# LeOPaRd: Towards Practical Post-Quantum Oblivious PRFs via 2HashDH Paradigm

Muhammed F. Esgin[1], Ron Steinfeld[1], Erkan Tairi[2⋆], and Jie Xu[1]

[1] Faculty of Information Technology, Monash University, Australia
[2] University of California, Berkeley
name.surname@monash.edu, erkan.tairi@berkeley.edu

**Abstract.** In this work, we introduce a more efficient post-quantum oblivious PRF (OPRF) design, called LeOPaRd. Our proposal is round-optimal and supports verifiability and partial obliviousness, all of which are important for practical applications. The main technical novelty of our work is a new method for computing samples of MLWE (Module Learning With Errors) in a two-party setting. To do this, we introduce a new family of (interactive) lattice problems, called *MLWE-PRF with re-use* (MLWE-PRF-RU). Here, the adversary is given a mix of MLWE and PRF samples where each PRF error is dependent on an *adversarially-chosen* matrix and the MLWE error. We rigorously study the hardness of MLWE-PRF-RU and provide a reduction from the standard MLWE to MLWE-PRF-RU, establishing a strong security foundation. We believe MLWE-PRF-RU problem family and the intermediate security reductions we introduce along the way can be of independent interest for other interactive protocols.
LeOPaRd exploits this MLWE-PRF-RU assumption to realize a lattice-based OPRF design without relying on heavy machinery such as noise flooding and fully homomorphic encryption used in earlier works. LeOPaRd can feature around 137 KB total communication, compared to 300+ KB in earlier works, while also achieving about 2× reduction in *online* communication compared to the prior state-of-the-art result. We also identify gaps in some existing constructions and models, and propose appropriate fixes.

**Keywords:** Oblivious PRF · Post-Quantum · Lattice · MLWE with Hints · 2HashDH

## 1 Introduction

An oblivious pseudorandom function (OPRF) extends a standard PRF in the following manner. In a standard PRF, a party in possession of a secret key $k$, can evaluate the PRF $F$, on any given input $x$ to obtain the output $y = F(k, x)$. The pseudorandomness of a PRF ensures infeasibility of distinguishing the outputs of this function from those of a random function. Now, in OPRF, the goal is to compute the PRF output $y$ in a two-party server-client model, where the server holds the secret key $k$ and the client has the input $x$. After they run the two-party OPRF protocol, we want the client to learn the PRF output $y$ without learning anything about the secret key $k$, and the server does not learn any information on the input $x$ or the output $y$. We say that an OPRF is *verifiable* OPRF (VOPRF) if the client is guaranteed that the output received is indeed evaluated under the committed key. As discussed in [CHL22, TCR+22], many applications of OPRFs such as for password-authenticated key exchange, checking compromised credentials and spam detection require a part of the client's input to be public to allow for domain separation for the PRF computation. In this case, we divide the client's input into a private part, $x$, and a public part, $t$, called the tag, and we call the resulting primitive a *partial* OPRF (POPRF).

(VP)OPRFs have become an important tool for privacy-preserving protocols and have numerous applications, including but not limited to private lightweight authentication mechanisms [DGS+18], private set intersection for checking compromised credentials [LPA+19, TPY+19], secure data de-duplication [KBR13], password-protected secret sharing [JKK14,JKKX16] and password-authenticated key exchange (PAKE) [JKX18].

⋆ Work done while the author was at ENS Paris.

| Scheme | Assumption | Rounds | Communication | Model |
|---|---|---|---|---|
| [ADDS21] | RLWE + SIS | 2 | >128GB | strong, QROM |
| [ADDG24] | heuristic | 2 | 15MB + 633KB | strong, ROM |
| [BDFH25] | Legendre PRF | 9 | 911KB | strong, ROM |
| [YBH+25] | Power Residue PRF | ≥5 | ≥1,600KB | strong, ROM |
| [AG24], $Q = 2^{16}$ | RLWE | 2 | 108KB + 189KB | weak, ROM |
| [AG24], $Q = 2^{32}$ | RLWE | 2 | 114KB + 198KB | weak, ROM |
| LeOPaRd, $Q_{x,t} = 2^{16}$ | MLWE-PRF-RU + MLWE + MSIS | 2 | (10 + 30)KB + 97KB | strong, ROM |
| LeOPaRd, $Q_{x,t} = 2^{32}$ (Section 4) | | 2 | (21 + 58)KB + 103KB | strong, ROM |

Table 1: Comparison of post-quantum (publicly) verifiable (P)OPRF constructions in the full malicious setting (without trusted setup). The summands in communication (when provided) indicate an offline cost and an online cost per query, respectively. For LeOPaRd, the offline cost is further broken down into the one-time (first term) and per-query (second term) costs while the offline cost is one-time for others (when provided). $Q$ denotes a *global* upperbound on the number of all OPRF queries; while $Q_{x,t}$ denotes the maximum number of OPRF queries *per fixed* PRF input/tag pair $(x, t)$. Enforcing the bound $Q_{x,t}$ is easy by counting the number of queries under the same *public* tag $t$.

Despite the widespread use of (VP)OPRFs, the existing constructions are either efficient but based on classical assumptions [FIPR05, JL09, JKK14, TCR+22], or based on (plausibly) post-quantum assumptions [ADDS21, ADDG24, BDFH25, AG24] but practically inefficient. Given the recent standardization efforts by the IETF[3] for the DH-based OPRFs of [JKK14, TCR+22] and by NIST for other post-quantum primitives[4], it is imperative to find efficient post-quantum OPRF constructions, which can pave the way for future post-quantum OPRF standards. We summarize the state of the art in post-quantum verifiable (P)OPRFs in Table 1 (extending a table from [AG24]). All schemes here provide security against malicious clients and servers (as needed in many applications). We also distinguish between the security models used in the schemes as "strong" if a stronger model as in [TCR+22] or UC framework is used[5], and as "weak" for otherwise weaker models.

### 1.1 Our Contributions

**Novel post-quantum VPOPRF construction.** Our first contribution in this work is a construction of a lattice-based VPOPRF, called LeOPaRd, that is more efficient than the current state-of-the-art. The comparison of our construction with the existing post-quantum VPOPRFs is given in Table 1, where we present our parameters for ≈ 95-bit security to match the security level for prior lattice-based schemes [AG24]. A more thorough performance analysis of our scheme is provided in Section 5, along with parameters for 128-bit security in Table 3. The efficiency gains of our construction come not only from utilizing the state-of-the-art lattice-based NIZK proof systems, such as LaBRADOR [BS23], but also by using a novel assumption that we introduce, and cleverly combining this with some additional lattice-based techniques that we explain in Section 1.3. The vast majority (∼ 90%) of the online communication cost for LeOPaRd in Table 1 arises from the underlying well-formedness NIZKs (performed by client and server). Excluding the NIZKs, LeOPaRd's core online OPRF communication cost is only around 0.5 KB for the server (vs 1.8 KB in [AG24]) and 7-13 KB for the client (vs ∼ 73 KB in [AG24]); see Tables 3 and 4 for more details.

We prove our construction secure in the random oracle model in Section 4.1. Specifically, we show that our construction achieves the strong security definitions from [TCR+22]: pseudorandomness, which provides

[3] https://datatracker.ietf.org/doc/rfc9497/

[4] https://csrc.nist.gov/projects/post-quantum-cryptography, https://csrc.nist.gov/projects/pqc-dig-sig

[5] We note that the UC model of OPRF is even stronger than the one given in [TCR+22].

security in the presence of malicious clients (POPRF security), and request privacy against malicious servers (POPRIV2 security). Moreover, in Appendix E.2 we show that our construction also achieves uniqueness, which in the verifiable setting ensures to the clients that the server honestly and consistently performs the blind evaluations. Furthermore, as described in Remark 4, our construction easily extends to $n$-out-of-$n$ threshold setting, as in [AG24], thanks to its key-homomorphic property.

**New MLWE-like assumption.** We introduce a family of (interactive) assumptions, called *Module Learning with Errors & PRF with Re-Use* (MLWE-PRF-RU), and use it to prove the security of our construction (against malicious clients). We believe introduction of such an interactive MLWE-like assumption is a natural next step given the interactive nature of protocols we deal with. The new assumption allows us not only to prove the security of our OPRF construction, but also to circumvent the shortcomings of the prior lattice-based OPRF constructions [ADDS21, ADDG24], such as removing the need for noise flooding or fully homomorphic encryption. We also note that our assumption is not "one-more" type (unlike earlier proposals, e.g. [JKK14, TCR+22, BDFH25]), and supports *adaptive* queries.

The MLWE-PRF-RU problem can be seen as a family of assumptions, which roughly state that a mix of MLWE and PRF samples are indistinguishable from uniform, even when the PRF samples have "deficient" errors that depend on *adversarially-chosen* (small-norm) matrices and MLWE errors. Crucially, to circumvent "trapdoor" attacks, the small matrix coefficients must be committed by the attacker *before* seeing the MLWE and PRF samples. This prevents the attacker from meaningfully biasing the "deficient" PRF samples. To establish strong confidence in the security of MLWE-PRF-RU, in Section 3, we provide a series of reductions that eventually prove that MLWE-PRF-RU is as hard as the standard MLWE problem (under appropriately chosen parameters). We note that the MLWE-PRF-RU assumption and the intermediate assumptions we introduce along the way might be of independent interest and find use in proving the security of other types of interactive protocols, such as blind signatures, as an alternative to one-more type assumptions.

**Issues with existing constructions and models.** As an additional contribution, we uncover flaws in the security proofs of two existing lattice-based OPRF constructions, [ADDS21] and [ADDG24]. We refer to Appendix E.1 for a more detailed exposition of these flaws and their potential fixes. We also observe that a reduction made in [TCR+22], that correctness and POPRIV2 security together imply uniqueness, does not necessary hold unless we have a one-to-one correspondence between the OPRF secret and public keys. In Appendix E.2, we patch this reduction by introducing a new property, called *key binding*, and show that correctness, key binding and POPRIV2 are sufficient for uniqueness. Along the way we also prove that LeOPaRd achieves key binding.

### 1.2 Related Work

The first post-quantum, and also lattice-based, VOPRF construction was given in [ADDS21], which although relied on a standard RLWE and SIS assumptions, was totally impractical as it required communication costs in the order of GBs. This was improved in [ADDG24] by combining fully homomorphic encryption with the weak PRF given in [BIP+18], but the reported bandwidth per query was in the order of MBs. Moreover, in order to achieve verifiability they needed to rely on a heuristic assumption about the hardness of evaluating deep circuits in an FHE scheme supporting only shallow circuits, which was recently broken [CJ25]. Recently, Beullens et al. [BDFH25] provided a VOPRF construction that is based on the Legendre PRF and which makes use of oblivious transfer (OT) and ZK proofs that can be instantiated from lattice assumptions, resulting also in a plausibly post-quantum secure VOPRF. However, this construction also has communication costs in the order of MBs.

**Comparison with concurrent works [AG24], [YBH+25], [HKL+25] and [DDT25].** In a concurrent and independent work published on IACR's ePrint just a few days before we uploaded the first version of this paper to ePrint, Albrecht and Gur [AG24] provided a new lattice-based ($n$-out-of-$n$) VOPRF construction. Their construction significantly improves, both assumption and performance wise, the earlier lattice-based VOPRF given in [ADDG24] as shown in Table 1. Similar to ours, their starting point is the OPRF construction given in [ADDS21], however, unlike us, they take a different technical route. They improve the construction given in [ADDS21] by first removing the reliance on the 1D-SIS assumption by borrowing a

trick from [GdKQ+24] and making use of Rényi divergence to avoid noise flooding, which together remove a superpolynomial factor from their modulus $q$. However, switching to Rényi divergence forbids them from using an indistinguishability-based security model, and in turn forces them to use a *weaker* security model than the one we use, which is the well-established OPRF security model given by Tyagi et al. [TCR+22]. Specifically, the model given in [TCR+22] is a simulation-based indistinguishability model that provides a careful and granular tracking of the oracle calls and comes close to the UC security model for blind protocols given in [JKK14]. On the other hand, the security model used by Albrecht and Gur [AG24] is akin to the one-more unforgeability definition used in the context of blind signatures, and is comparatively weaker than the model of Tyagi et al. [TCR+22]. Moreover, the use of Rényi divergence forces them to use a *global* bound on number of all OPRF queries, which can potentially hinder the practicality. Our proposal, on the other hand, relies on a bound $Q_{x,t}$ on the number of queries *per input/tag pair* $(x, t)$. Given the tag $t$ is *public*, it is straightforward for the server to count the number of OPRF queries made per tag $t$. For example, if the tag is a fixed username (unique per client) as in PAKE protocols [JKX18], we can set $Q_{x,t} = 2^{24}$ *per client* and support a practically unlimited number of clients, say, $2^{64}$; whereas [AG24] would need to set $Q = 2^{88}$ to support the same setting, significantly degrading efficiency. Note that exhausting $Q_{x,t} = 2^{24}$ queries would require a client to continuously make queries *every minute* for about *32 years* (1 year $\approx 2^{19}$ minutes).

Apart from the aforementioned improvements over the prior work [ADDS21], Albrecht and Gur [AG24] also make use of the newer lattice-based proof systems, specifically LNP22 [LNP22] and LaBRADOR [BS23], analogous to us. However, in order for their security proof to go through, they require the underlying proof system to be straight-line extractable. This can be done for LNP22 using the Katsumata transform [Kat21] by incurring a significant (4-8×) performance penalty [NO25]. However, it is an open problem how to obtain straight-line extractability for LaBRADOR. Importantly, we do not require straight-line extractability from the underlying NIZK proof systems for our security proofs. Lastly, the additional overhead that comes from the straight-line extractability, along with the fact that they aim only for 90-100 bit security level, are not properly reflected in their reported numbers, which are also shown in Table 1. Given that they [AG24] rely on (fully structured) Ring LWE problem, they have less flexibility in choosing their parameters, and would need to incur almost 2× performance penalty if their parameters aimed at 128-bit or higher security level. To match the security level of [AG24], our reported results in Table 1 also aim at 90-95 bits of security, and we provide parameters and performance results at 128-bit security level in Table 3. Overall, accounting for (i) a stronger security model, (ii) straight-line extractability for NIZKs, and (iii) at least 128-bit security level would lead to a significant cost increase in [AG24]. Our proposal LeOPaRd and its performance analysis readily account for all these features and even partial obliviousness.

Another concurrent work, Gold OPRF [YBH+25], takes a completely different technical approach, using a generalization of Legendre PRF, namely Power Residue PRFs. In the full malicious setting (without trusted setup), their communication cost (without amortization) is significantly higher than ours ($> 10\times$) while also requiring at least 5 rounds of communication (compared to our round-optimal proposal). On the other hand, Gold OPRF can support efficient batching on both the server and client sides, but the amortization starts to make a significant effect only after batch sizes above thousands, which may not be realistic in real-life applications. Moreover, Gold OPRF primarily targets the pre-processing model, which does not fit some important applications of OPRFs exhibiting a password-only setting such as password-authenticated key exchange (PAKE) protocols. Our LeOPaRd proposal, however, can fit such password-only settings.

Another concurrent work, LEAP, in [HKL+25] is considered only in the semi-honest model for both the client and server (therefore, also is not included in Table 1). Even in this model, it requires 6 rounds of communication, 23 KB online communication, and almost 800 KB of preprocessing communication. Moreover, it is based on a heuristic security assumption, and we are aware that LEAP is actually insecure [Hei25].

In a very recent work, Davidson et al. [DDT25] extended LEAP [HKL+25] by removing its heuristic assumption, and achieving round-optimal online phase by utilizing preprocessing. However, they also only achieve semi-honest security (and hence, is not included in Table 1). Their online communication is 11.9 KB, but the preprocessing requires between 790 KB and 1.9 MB communication depending on parameter choices (as highlighted in [DDT25, Table 7]).

### 1.3 Technical Overview

**VPOPRF construction.** The 2HashDH OPRF paradigm [JKK14] (and its extension to partially oblivious setting in [TCR+22]) has been shown to be highly effective in the discrete-log setting. Therefore, our goal is to efficiently realize this high-level paradigm in the lattice setting. An initial attempt was already done in [ADDS21], which serves as a starting point for our scheme. For simplicity, our discussion here focuses on the regular OPRF setting (without partial obliviousness).

The high-level idea of the 2HashDH paradigm is to first hash the PRF input $x$ to some group element $b_x = \mathsf{H}_1(x)$. This term is then blinded via a randomness $r$ and the resulting value $c_x = \mathsf{H}_1(x)^r$ is sent to the server. The server then blindly performs PRF evaluation using its key $k$ as $u_x = c_x^k$. The blinded evaluation result $u_x$ is sent to the client, who unblinds the value using their randomness $r$ to obtain a PRF value $z = \mathsf{H}_1(x)^k$. The final PRF output is computed via a second hashing as $y = \mathsf{H}_2(x, z)$. The blind/unblind operations rely on the homomorphic properties of the computations performed in the protocol.

To realize this idea in the lattice setting over a ring $R_q$ with modulus $q$, we want to encrypt the matrix $\mathbf{B}_x = \mathsf{H}(x)$ using homomorphic encryption. We can do this as follows.

1. Client computes $\mathbf{C}_x = \mathbf{R}\mathbf{A}_r + \mathbf{B}_x$ for randomness $\mathbf{R}$ and public matrix $\mathbf{A}_r$.
2. Upon receiving $\mathbf{C}_x$, the server can compute $\mathbf{u}_x = \mathbf{C}_x\mathbf{k} + \mathbf{e}'_s$ using its key $\mathbf{k}$ with some error vector $\mathbf{e}'_s$. Observe that $\mathbf{u}_x = \mathbf{R}\mathbf{A}_r\mathbf{k} + \mathbf{B}_x\mathbf{k} + \mathbf{e}'_s$.
3. Given $\mathbf{u}_x$ and additionally $\mathbf{v}_k = \mathbf{A}_r\mathbf{k} + \mathbf{e}_s$, the client can compute $\mathbf{u}_x - \mathbf{R}\mathbf{v}_k = \mathbf{B}_x\mathbf{k} + \mathbf{e}_f$ for some final error term $\mathbf{e}_f$.

Provided the final error is small enough, the client can round it off to arrive at $\lfloor \mathbf{B}_x\mathbf{k} \rceil_p$ where $\lfloor \cdot \rceil_p$ denotes multiplying each coefficient by $p/q$ and rounding to the nearest integer. Both parties additionally use NIZK proofs to show the correctness of their computations to provide security and verifiability. What we have discussed so far is effectively the high-level blueprint in [ADDS21].

The main technical difficulty in realizing this idea *efficiently* is that we need to argue that $\mathbf{u}_x$ does not leak information about the server's key $\mathbf{k}$. Observe that even though $\mathbf{u}_x$ looks like an MLWE sample, the matrix $\mathbf{C}_x$ is controlled by the (malicious) client, and therefore, is not guaranteed to be uniformly random. To get around this problem, [ADDS21] relies on noise flooding (a.k.a. smudging) with an exponentially large error $\mathbf{e}'_s$ with coefficients of size $O(2^\lambda)$ to make sure that no secret information is leaked. However, then the final error term that we need to get rid of is also of size $O(2^\lambda)$, meaning that we need $q = O(2^{2\lambda})$ to arrive at the correct PRF output with overwhelming probability. Noting that the overall communication size is *quadratic* in $\log q$, such a large modulus leads to an inefficient construction[6].

Beyond using state-of-the-art NIZK proofs in our scheme, the main technical novelty of our LeOPaRd proposal is that we introduce a new method to argue server security *without* relying on noise flooding. In particular, we want to force the client into committing to its randomness $\mathbf{R}$ *before* seeing the public matrix $\mathbf{A}_r$. To this end, the client first commits to $\mathbf{R}$, resulting in $\mathbf{c}_r$, and uses a hash function (modeled as a random oracle) to derive the matrix $\mathbf{A}_r$. One can immediately observe that the client's control in $\mathbf{C}_x$ is now significantly limited and intuitively revealing $\mathbf{u}_x$ as above should be secure based on the randomness of $\mathbf{A}_r$ and the fact that the client in any case obtains a noisy vector close to $\mathbf{B}_x\mathbf{k}$ in their final step.

**Formal security analysis via MLWE-PRF-RU assumption.** We go beyond intuitive thinking and analyze the security of the construction rigorously. The client's commitment $\mathbf{c}_r$ to $\mathbf{R}$ can be seen as the client submitting an oracle query with input $\mathbf{R}$ to obtain back a random matrix $\mathbf{A}_r$. To realize this in the security reduction while adhering to client security, we use an *extractable* commitment scheme that enables extraction/decryption of the committed message using a trapdoor (that the reduction knows in the security analysis). We then use our new assumption, MLWE-PRF-RU, to prove the pseudorandomness property (server security) of LeOPaRd. In MLWE-PRF-RU, the adversary is given access to (SampMLWEcom, SampPRF-RU) oracles that, on input $(\mathbf{R}, x)$ returns a random matrix $\mathbf{A}_r$, an MLWE sample $\mathbf{c}$ under $\mathbf{A}_r$, and a PRF-like

[6] We note here that [ADDS21] also used earlier NIZK proof systems, which are very inefficient compared to more state-of-the-art proofs of today. However, even without considering NIZKs, [ADDS21] incurs communication in the order of MBs.

$$\boxed{\mathsf{MLWE\text{-}PRF\text{-}RU}: \quad \mathbf{c}_i = \mathbf{A}_i\mathbf{k} + \mathbf{e}_i, \quad \mathbf{z}_{x,i} = \mathbf{B}_x\mathbf{k} + \mathbf{e}'_i - \mathbf{R}_i\mathbf{e}_i \quad : \quad \mathbf{e}_i \xleftarrow{\$} \chi^N, \quad \mathbf{e}'_i \xleftarrow{\$} \chi_1^h}$$

$$\equiv$$

$$\boxed{\mathbf{c}_i = \mathbf{A}_i\mathbf{k} + \mathbf{e}_i, \quad \mathbf{z}_{x,i} = \mathbf{B}_x\mathbf{k} + \mathbf{e}'_{B,i} + \bar{\mathbf{e}}'_i - \mathbf{R}_i\mathbf{e}_i \quad : \quad \mathbf{e}'_{B,i} \xleftarrow{\$} \chi_{1,B}^h, \quad \bar{\mathbf{e}}'_i \xleftarrow{\$} \bar{\chi}_1^h, \quad \chi_1 = \chi_{1,B} + \bar{\chi}_1}$$

Section 3.1; Theorem 1 $\geq$

$$\boxed{\mathsf{Hint\text{-}MLWE\text{-}PRF}: \quad \mathbf{c}_i = \mathbf{A}_i\mathbf{k} + \mathbf{e}_i, \quad \mathbf{y}_{B,x} = \mathbf{B}_x\mathbf{k} + \mathbf{e}_{B,x}, \quad \mathbf{h}_{B,i} = \mathbf{e}'_{B,i} - \mathbf{e}_{B,x}, \quad \mathbf{h}_i = \bar{\mathbf{e}}'_i - \mathbf{R}_i\mathbf{e}_i \quad : \quad \mathbf{e}_{B,x} \xleftarrow{\$} \chi_B^h}$$

Section 3.2; Theorem 2 $\geq$ (via extension of Matrix-Hint-MLWE [KLSS23, ENP24])

$$\boxed{\mathsf{MLWE\text{-}PRF}: \quad \mathbf{c}_i = \mathbf{A}_i\mathbf{k} + \mathbf{e}_i, \quad \mathbf{y}_{B,x} = \mathbf{B}_x\mathbf{k} + \mathbf{e}_{B,x} \quad : \quad \mathbf{e}_i \xleftarrow{\$} \chi_0^N, \quad \mathbf{e}_{B,x} \xleftarrow{\$} \chi_{0,B}^h}$$

Section 3.3; Theorems 3 and 4 $\geq$ (using random oracle or BP14 approach [BP14])

$$\boxed{\mathsf{MLWE}: \quad \mathbf{c}_i = \mathbf{A}_i\mathbf{k} + \mathbf{e}_i \quad : \quad \mathbf{e}_i \xleftarrow{\$} \chi_0^N}$$

Fig. 1: High-level (simplified) summary of our reductions with changes marked in gray boxes. We denote by "$\equiv$" equivalence, and by "$\geq$" reduction. In all experiments (where relevant), $(\mathbf{R}_i, x)$ is an (adaptively chosen) input by an adversary, and $\mathbf{A}_i \xleftarrow{\$} R_q^{N\times m}$; $\mathbf{k} \xleftarrow{\$} \chi_k^m$ and $\mathbf{B}_x := \mathsf{H}(x)$ for a "message mapping" $\mathsf{H}$.

sample $\mathbf{z}$ that the client is able to derive in the OPRF protocol. Given these values, the term $\mathbf{u}_x$ above is uniquely determined as $\mathbf{u}_x := \mathbf{z} + \mathbf{R}\mathbf{v}_k$. The MLWE error is re-used in both $\mathbf{c}$ and $\mathbf{z}$, and hence the naming of the assumption. The client security is easier to argue and is proven based on MLWE.

**Security reduction for MLWE-PRF-RU from MLWE-PRF and MLWE.** We introduce a series of reductions to establish strong security foundations for MLWE-PRF-RU as depicted in Figure 1. First, we aim to get around the deficiency in the PRF samples and move to a more standard assumption, we call MLWE-PRF, where both MLWE and PRF samples have *independent* and proper small errors following a Gaussian distribution. That is, the MLWE-PRF assumption is that the randomized function $x \mapsto \mathsf{H}(x) \cdot \mathbf{k} + \mathbf{e}$ with small Gaussian error $\mathbf{e}$ is pseudorandom even when given additional MLWE samples for $\mathbf{k}$ with respect to uniformly random matrices $\mathbf{A}$. This MLWE-PRF $\rightarrow$ MLWE-PRF-RU reduction step is the main technically challenging part that we elaborate on further below. The latter step of arguing security of MLWE-PRF is easier. First of all, if we model $\mathsf{H}$ as a random oracle, then the (randomized) PRF samples simply become MLWE samples, resulting immediately in a reduction from MLWE to MLWE-PRF, and hence to our main MLWE-PRF-RU assumption. This result demonstrates that the only way an attacker can break MLWE-PRF is by exploiting the structure of the "message mapping" $\mathsf{H}$ (or breaking standard MLWE). We therefore consider other natural instantiations of $\mathsf{H}$ as in earlier lattice-based PRF proposals [BLMR13, BP14], which already proved that certain algebraic instantiations of $\mathsf{H}$ still maintains pseudorandomness based on MLWE. We extend on the results in [BP14] to base the hardness of MLWE-PRF-RU on the standard MLWE problem (see Section 3.3).

Let us now discuss the MLWE-PRF to MLWE-PRF-RU reduction at a high level. In the MLWE-PRF-RU problem[7], each of the attacker's oracle queries $(x, \mathbf{R}_i)$ is answered with MLWE-PRF samples $(\mathbf{A}_i, \mathbf{c}_i = \mathbf{A}_i\mathbf{k} + \mathbf{e}_i, \mathbf{z}_{x,i} = \mathbf{B}_x\mathbf{k} + \mathbf{e}'_i - \mathbf{R}_i\mathbf{e}_i)$ where $\mathbf{B}_x := \mathsf{H}(x)$ and $\mathbf{e}_j, \mathbf{e}'_j$ are some Gaussian errors, reminiscent of the Hint-MLWE problem [KLSS23]. Accordingly, our security reduction from MLWE-PRF to MLWE-PRF-RU proceeds in two steps. In the first step, we reduce a variant of the Hint-MLWE problem called Hint-MLWE-PRF to

[7] Our description here is simpler than the actual MLWE-PRF-RU assumption in Section 3. In the real definition, we consider even a stronger adversary that can adaptively request PRF samples after seeing the MLWE samples.

MLWE-PRF-RU. In the Hint-MLWE-PRF problem, besides the MLWE-like samples w.r.t. $\mathbf{B}_x$ and $\mathbf{A}_j$ matrices (i.e., $\mathbf{y}_B = \mathbf{B}_x\mathbf{k} + \mathbf{e}_B$ and $\mathbf{c}_j = \mathbf{A}_j\mathbf{k} + \mathbf{e}_j$ as before), the adversary is also provided with appropriate linear combinations of the error vectors as hints of the form $\mathbf{h}_j := \mathbf{e}'_j - \mathbf{R}_j\mathbf{e}_j$ and $\mathbf{h}_{B,j} := \mathbf{e}'_{B,j} - \mathbf{e}_B$. These hints allow the reduction from MLWE-PRF to MLWE-PRF-RU to simulate the 'deficient' PRF samples as $\mathbf{z}_{x,j} := \mathbf{y}_B + \mathbf{h}_{B,j} + \mathbf{h}_j = \mathbf{B}_x\mathbf{k} + \bar{\mathbf{e}}'_j - \mathbf{R}_j\mathbf{e}_j$, where $\bar{\mathbf{e}}'_j := \mathbf{e}'_j + \mathbf{e}'_{B,j}$ is the simulated error term for $\mathbf{z}_{x,j}$.[8] Note that in Hint-MLWE-PRF each $\mathbf{h}_j$ hint contains a *linear combination of several* error vectors (rather than hints containing only *scalar multiples* of the error vector as in [KLSS23]). Hence Hint-MLWE-PRF can be viewed as a variant of the *Matrix* Hint-MLWE [ENP24] that generalizes Hint-MLWE [KLSS23] to the case of hint matrices $\mathbf{R}_j$. Our Hint-MLWE-PRF variant further includes a HintPRF oracle that provides the PRF samples $\mathbf{y}_B$ along with the hints $\mathbf{h}_{B,j}$ on the PRF error. In the second step, we present a reduction from MLWE-PRF to Hint-MLWE-PRF by extending the MLWE to Matrix Hint-MLWE [ENP24] reduction.

Overall, we achieve a reduction from MLWE to MLWE-PRF-RU, provided that the necessary parameter constraints are satisfied. Our MLWE to MLWE-PRF-RU reduction is essentially tight in the sense that our reduction's parameter condition on the underlying error width parameter is *necessary* to prevent an existing *attack* on MLWE-PRF-RU. In particular, our reduction requires the "masking ratio" lower bound $\sigma_1/\sigma \geq \sqrt{Q_x^\infty}$ for large $Q_x^\infty$, where $\sigma_1$ and $\sigma$ denote the width (std dev) of the masking errors $\mathbf{e}'_j$ and MLWE errors $\mathbf{e}_j$, respectively, and $Q_x^\infty$ denotes an upper bound on the oracle samples queried by the adversary *per input* $x$ (corresponding to input/tag pair $(x,t)$ in our POPRF protocol). On the other hand, this lower bound is necessary to protect against a simple error "averaging" attack on MLWE-PRF-RU (and our OPRF protocol as well as those using the same blueprint [ADDS21, AG24]), which we describe in Appendix B.

## 2 Preliminaries

**Notation.** We denote the security parameter by $\lambda \in \mathbb{N}$, and $\kappa \in \mathbb{N}$ denotes the statistical correctness parameter. We will aim to obtain the correct PRF value except with probability at most $2^{-\kappa}$. For $n \in \mathbb{N}$, set $[n] = \{1, 2, \ldots, n\}$. We denote by $x \xleftarrow{\$} X$ the uniform sampling of the variable $x$ from the set $X$, and we denote the uniform distributions on a set $X$ by $\mathcal{U}(X)$. We write $x \leftarrow \mathsf{A}(y)$ to denote that a probabilistic polynomial time (PPT) algorithm $\mathsf{A}$ on input $y$, outputs $x$. When we want to make the randomness $r$ used by $\mathsf{A}$ explicit, we write it as $x \leftarrow \mathsf{A}(y;r)$. If $\mathsf{A}$ has oracle access to a procedure $\mathsf{O}$, we represent this by superscript $\mathsf{A}^{\mathsf{O}}$. If $\mathsf{A}$ is a deterministic polynomial time (DPT) algorithm, we use the notation $x := \mathsf{A}(y)$. We use the same notation for the projection of tuples, e.g., we write $\sigma := (\sigma_1, \sigma_2)$ for a tuple $\sigma$ composed of two elements $\sigma_1$ and $\sigma_2$. We define *polynomial* functions as $\mathsf{poly}(\lambda) = \bigcup_{d\in\mathbb{N}} O(\lambda^d)$ and *negligible* functions as $\mathsf{negl}(\lambda) = \bigcap_{d\in\mathbb{N}} o(\lambda^{-d})$. We write the convolution of two distributions $D_1$ and $D_2$ as $D_1 + D_2$.

We denote by $R_q = \mathbb{Z}_q[X]/(X^d + 1)$. We write $R_{\mathsf{bin}}$ to denote the set of polynomials in $R_q$ that have binary coefficients. For $a \in \mathbb{Z}^+$, we use $\mathbb{S}_a$ to denote the set of polynomials in $R_q$ with infinity norm at most $a$. The rounding operation $\lfloor a \rceil_p : R_q \to R_p$ is defined as multiplying $a$ by $p/q$ and rounding each resulting coefficient to the nearest integer. When $\mathbf{a}$ is a vector, then by $\lfloor \mathbf{a} \rceil_p$ we consider coordinate-wise rounding. For a matrix $\mathbf{R}$, we denote by $\|\mathbf{R}\|$, $\|\mathbf{R}\|_1$ and $\|\mathbf{R}\|_{\infty,M}$ its matrix 2-norm (largest singular value), matrix 1-norm and matrix $\infty$-norm, respectively and by $\sigma_{\min}(\mathbf{R})$ and $\sigma_{\max}(\mathbf{R})$ the smallest (resp. largest) singular values of $\mathbf{R}$. We denote by $\|\mathbf{R}\|_\infty$ the entry-wise maximum absolute value of $\mathbf{R}$.

As the NIZK relations to be proven get more complex for various protocols, we believe it is imperative to have a more explicit and concise notation. We introduce the notation "$\mathfrak{D}(x); \mathfrak{K}(w)$" to describe a NIZK relation to mean "given a statement $x$ and knowledge of witness $w$".

Next, we provide some of the preliminaries needed for the rest of the paper, and refer the reader to Appendix A for additional preliminaries.

[8] We remark that this reduction could also directly use a single combined hint of the form $\bar{\mathbf{h}}_j := \mathbf{h}_{B,j} + \mathbf{h}_j = \bar{\mathbf{e}}'_j - \mathbf{R}_j\mathbf{e}_j - \mathbf{e}_B$. However, such a combined variant of Hint-MLWE makes it difficult to deal with *adaptive* oracle $\mathbf{R}_j$ queries in the second step of reducing from MLWE-PRF to our Hint-MLWE-PRF problem. Hence, we split the hint into the two separate hints $\mathbf{h}_j$ and $\mathbf{h}_{B,j}$.

### 2.1 (Partially) Oblivious PRF

We use a slightly modified version of the game-based definitions for (round-optimal) OPRF with preprocessing given in [DDT25], which in turn extends the definition given in [TCR+22]. An OPRF is a protocol between a server $\mathsf{S}$ that has a private key $k$ and a client $\mathsf{C}$ that wants to obtain an evaluation of PRF $F_k$ on inputs of its choice. We say that an OPRF is a partial OPRF (POPRF) if part of the client's input is given to the server, i.e., $y = F_k(t, x)$, where $t$ is in the public part and $x$ is the private part hidden from $\mathsf{S}$. When the client can verify the correctness and consistency of the PRF evaluations, then we consider verifiable OPRF (VOPRF).

**Definition 1 (Partial Oblivious PRF with Preprocessing).** *A partial oblivious PRF (POPRF)* $\mathsf{F}$ *is a tuple of* PPT *algorithms*

$$(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{PreProcClient}, \mathsf{PreProcServer}, \mathsf{Request}, \mathsf{BlindEval}, \mathsf{Finalize}, \mathsf{Eval}).$$

*The* Setup *and* KeyGen *algorithms generate public parameters* pp *and a public/secret key pair* $(\mathsf{pk}, \mathsf{sk})$*, respectively. The preprocessing is a (single-round) protocol between the client* C *and server* S*, presented as algorithms* F.PreProcClient *and* F.PreProcServer*, which work as follows (where the random oracles are denoted by* RO*):*

1. C *runs the algorithm* $\mathsf{F.PreProcClient}^{\mathsf{RO}}_{\mathsf{pp}}(\mathsf{pk}, T)$ *taking as input a public key* pk *and an evaluation bound* $T$*. It outputs a local client state* $\mathsf{st_C}$ *and a preprocessing request message* preq*, which is sent to* S*.*
2. S *runs* $\mathsf{F.PreProcServer}^{\mathsf{RO}}_{\mathsf{pp}}(\mathsf{sk}, \mathsf{preq})$ *taking as input a secret key* sk *and the preprocessing request message* preq*. It produces a local server state* $\mathsf{st_S}$ *and a preprocessing response message* prep*, which is sent to* C*. (Without loss of generality we assume that the client* C *includes the evaluation bound* $T$ *and response message* prep *in the state* $\mathsf{st_C}$*.)*

*Oblivious evaluation is carried out as a (single-round) protocol between* C *and* S*, presented as algorithms* F.Request, F.BlindEval, F.Finalize*, which work as follows:*

1. *First,* C *runs the algorithm* $\mathsf{F.Request}^{\mathsf{RO}}_{\mathsf{pp}}(\mathsf{pk}, t, x, \mathsf{st_C})$ *taking as input a public key* pk*, a (public) tag* $t$*, a private input* $x$ *and previously constructed client state* $\mathsf{st_C}$*. It outputs an updated client state* $\mathsf{st_C}$ *and a request message* req *that is sent to* S*, or outputs* $\perp$ *(a designated error symbol).*
2. S *runs* $\mathsf{F.BlindEval}^{\mathsf{RO}}_{\mathsf{pp}}(\mathsf{sk}, t, \mathsf{req}, \mathsf{st_S})$ *taking as input a secret key* sk*, a (public) tag* $t$*, the request message* req *and previously constructed server state* $\mathsf{st_S}$*. It produces a response message* rep*, which is sent to* C*.*
3. *Finally,* C *runs* $\mathsf{F.Finalize}^{\mathsf{RO}}_{\mathsf{pp}}(\mathsf{rep}, \mathsf{st_C})$ *taking as input the response message* rep *and the previously constructed client state* $\mathsf{st_C}$*. It outputs either a PRF evaluation* $y$ *or* $\perp$ *if* rep *is rejected.*

*The unblinded evaluation algorithm* F.Eval *is deterministic and takes as input a secret key* sk*, an input pair* $(t, x)$*, and outputs a PRF evaluation* $y$*.*

*We also define the sets* $\mathcal{SK}, \mathcal{PK}, \mathcal{T}, \mathcal{X}, \mathcal{Z}$*, representing the secret key, public key, (public) tag, private input, and output spaces, respectively. We define the input space as* $\mathcal{T} \times \mathcal{X}$*. We assume there exist efficient algorithms for sampling and membership queries on these sets.*

We remark that fixing $t = \perp$, gives us the definition of (plain) OPRF. We extend the correctness definition from [ADDG24], which allows for a small failure probability.

**Definition 2 (Correctness).** *A partial oblivious PRF (POPRF)* $\mathsf{F}$ *is correct, if for all* $\lambda, T \in \mathbb{N}$*, for all* $\mathsf{pp} \leftarrow \mathsf{F.Setup}(1^\lambda)$ *and* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{F.KeyGen}^{\mathsf{RO}}_{\mathsf{pp}}(1^\lambda)$*, it holds that*

$$\Pr\left[ y = \mathsf{F.Eval}^{\mathsf{RO}}_{\mathsf{pp}}(\mathsf{sk}, t, x) \;\middle|\; \begin{array}{r} (\mathsf{st_C}, \mathsf{preq}) \leftarrow \mathsf{F.PreProcClient}^{\mathsf{RO}}_{\mathsf{pp}}(\mathsf{pk}, T) \\ (\mathsf{st_S}, \mathsf{prep}) \leftarrow \mathsf{F.PreProcServer}^{\mathsf{RO}}_{\mathsf{pp}}(\mathsf{sk}, \mathsf{preq}) \\ (\mathsf{st_C}, \mathsf{req}) \leftarrow \mathsf{F.Request}^{\mathsf{RO}}_{\mathsf{pp}}(\mathsf{pk}, t, x, \mathsf{st_C}) \\ \mathsf{rep} \leftarrow \mathsf{F.BlindEval}^{\mathsf{RO}}_{\mathsf{pp}}(\mathsf{sk}, t, \mathsf{req}, \mathsf{st_S}) \\ y \leftarrow \mathsf{F.Finalize}^{\mathsf{RO}}_{\mathsf{pp}}(\mathsf{rep}, \mathsf{st_C}) \end{array} \right] = 1 - \mathsf{negl}(\lambda).$$

$\mathsf{POPRF}^b_{\mathsf{F},\mathcal{A},\mathcal{S},\mathsf{RO}}(\lambda)$

$q_p \coloneqq 0, \mathcal{D} \coloneqq \bot$
$\mathsf{st}_\mathsf{F} \leftarrow \mathsf{RO}_\mathsf{F}.\mathsf{Init}(1^\lambda)$ // $\mathcal{T} \times \mathcal{X} \to \mathcal{Z}$
$\mathsf{st}_\mathsf{RO} \leftarrow \mathsf{RO}.\mathsf{Init}(1^\lambda)$
$\mathsf{pp}_1 \leftarrow \mathsf{F}.\mathsf{Setup}(1^\lambda)$
$(\mathsf{st}_\mathcal{S}, \mathsf{pk}_0, \mathsf{pp}_0) \leftarrow \mathcal{S}.\mathsf{Init}(\mathsf{pp}_1)$
$(\mathsf{sk}, \mathsf{pk}_1) \leftarrow \mathsf{F}.\mathsf{KeyGen}^{\mathsf{RO}}_{\mathsf{pp}_1}(1^\lambda)$
$b' \leftarrow \mathcal{A}^{\mathsf{PreProcServer},\mathsf{Eval},\mathsf{BlindEval},\mathsf{Prim}}(\mathsf{pp}_b, \mathsf{pk}_b)$
**return** $b'$

$\mathsf{BlindEval}(i, t, \mathsf{req})$

**if** $i \notin [q_p]$ **then return** $\bot$
**parse** $\mathcal{D}[i] \coloneqq \mathsf{st}_\mathsf{S}$
$(\mathsf{rep}_0, \mathsf{st}_\mathcal{S}) \leftarrow \mathcal{S}.\mathsf{BlindEval}(t, \mathsf{req}, \mathsf{st}_\mathcal{S})$
$\mathsf{rep}_1 \leftarrow \mathsf{F}.\mathsf{BlindEval}^{\mathsf{RO}}_{\mathsf{pp}_1}(\mathsf{sk}, t, \mathsf{req}, \mathsf{st}_\mathsf{S})$
**return** $\mathsf{rep}_b$

$\mathsf{PreProcServer}(\mathsf{preq})$

$q_p \coloneqq q_p + 1$
$(\mathsf{prep}_0, \mathsf{st}_\mathcal{S}) \leftarrow \mathcal{S}.\mathsf{PreProcServer}(\mathsf{preq}, \mathsf{st}_\mathcal{S})$
$(\mathsf{st}_\mathsf{S}, \mathsf{prep}_1) \leftarrow \mathsf{F}.\mathsf{PreProcServer}^{\mathsf{RO}}_{\mathsf{pp}_1}(\mathsf{sk}, \mathsf{preq})$
$\mathcal{D}[q_p] \coloneqq \mathsf{st}_\mathsf{S}$
**return** $\mathsf{prep}_b$

$\mathsf{Eval}(t, x)$

$z_0 \leftarrow \mathsf{RO}_\mathsf{F}.\mathsf{Eval}((t, x), \mathsf{st}_\mathsf{F})$
$z_1 \leftarrow \mathsf{F}.\mathsf{Eval}^{\mathsf{RO}}_{\mathsf{pp}_1}(\mathsf{sk}, t, x)$
**return** $z_b$

$\mathsf{Prim}(x)$

$(h_0, \mathsf{st}_\mathcal{S}) \leftarrow \mathcal{S}.\mathsf{Eval}(x, \mathsf{st}_\mathcal{S})$
$h_1 \leftarrow \mathsf{RO}.\mathsf{Eval}(x, \mathsf{st}_\mathsf{RO})$
**return** $h_b$

$\mathsf{POPRIV2}^b_{F,\mathcal{A},\mathsf{RO}}(\lambda)$

$q_r \coloneqq 0, q_p \coloneqq 0, \mathcal{D}_p \coloneqq \mathcal{D}_r \coloneqq \bot$
$\mathsf{pp} \leftarrow \mathsf{F}.\mathsf{Setup}(1^\lambda)$
$b' \leftarrow \mathcal{A}^{\mathsf{PreProcClient},\mathsf{Request},\mathsf{Finalize},\mathsf{RO}}(\mathsf{pp})$
**return** $b'$

$\mathsf{Request}(i, \mathsf{pk}, \mathsf{prep}, t, x_0, x_1)$

**if** $i \notin [q_p]$ **then return** $\bot$
**parse** $\mathcal{D}_p[i] \coloneqq \mathsf{st}_\mathsf{C}$
$\mathsf{st}_\mathsf{C} \coloneqq \mathsf{st}_\mathsf{C} \cup \{\mathsf{prep}\}$
$(\mathsf{st}_{\mathsf{C},q_r,0}, \mathsf{req}_0) \leftarrow \mathsf{F}.\mathsf{Request}^{\mathsf{RO}}_{\mathsf{pp}}(\mathsf{pk}, t, x_0, \mathsf{st}_\mathsf{C})$
$(\mathsf{st}_{\mathsf{C},q_r,1}, \mathsf{req}_1) \leftarrow \mathsf{F}.\mathsf{Request}^{\mathsf{RO}}_{\mathsf{pp}}(\mathsf{pk}, t, x_1, \mathsf{st}_\mathsf{C})$
**if** $\mathsf{req}_0 = \bot \vee \mathsf{req}_1 = \bot$ **then**
  **return** $\bot$
$q_r \coloneqq q_r + 1$
$\mathcal{D}_r[i, q_r] \coloneqq (\mathsf{st}_{\mathsf{C},q_r,0}, \mathsf{st}_{\mathsf{C},q_r,1})$
**return** $(\mathsf{req}_b, \mathsf{req}_{1-b})$

$\mathsf{PreProcClient}(\mathsf{pk}, T)$

$q_p \coloneqq q_p + 1$
$(\mathsf{st}_\mathsf{C}, \mathsf{preq}) \leftarrow \mathsf{F}.\mathsf{PreProcClient}^{\mathsf{RO}}_{\mathsf{pp}}(\mathsf{pk}, T)$
$\mathcal{D}_p[q_p] \coloneqq \mathsf{st}_\mathsf{C}$
**return** $\mathsf{preq}$

$\mathsf{Finalize}(i, j, \mathsf{rep}_0, \mathsf{rep}_1)$

**if** $i \notin [q_p] \ \vee \ j \notin [q_r]$ **then**
  **return** $\bot$
**parse** $\mathcal{D}_r[i, j] \coloneqq (\mathsf{st}_{\mathsf{C},j,0}, \mathsf{st}_{\mathsf{C},j,1})$
$y_b \leftarrow \mathsf{F}.\mathsf{Finalize}^{\mathsf{RO}}_{\mathsf{pp}}(\mathsf{rep}_0, \mathsf{st}_{\mathsf{C},j,b})$
$y_{1-b} \leftarrow \mathsf{F}.\mathsf{Finalize}^{\mathsf{RO}}_{\mathsf{pp}}(\mathsf{rep}_1, \mathsf{st}_{j,1-b})$
**if** $y_0 = \bot \vee y_1 = \bot$ **then**
  **return** $\bot$
**return** $(y_0, y_1)$

Fig. 2: POPRF and POPRIV2 experiments.

Next, we extend the *pseudorandomness* against malicious clients definition from [TCR+22] to account for preprocessing.

**Definition 3 (Pseudorandomness).** *We say that a partial oblivious PRF (POPRF)* $\mathsf{F}$ *is pseudorandom if for all (stateful)* PPT *adversaries* $\mathcal{A}$*, there exists a* PPT *simulator* $\mathcal{S}$ *and a negligible function* $\mathsf{negl}(\cdot)$*, such that the following holds,*

$$\mathsf{Adv}^{\mathsf{po\text{-}prf}}_{\mathsf{F},\mathcal{A},\mathcal{S},\mathsf{RO}}(\lambda) = \left|\Pr\left[\mathsf{POPRF}^1_{\mathsf{F},\mathcal{A},\mathcal{S},\mathsf{RO}}(\lambda) = 1\right] - \Pr\left[\mathsf{POPRF}^0_{\mathsf{F},\mathcal{A},\mathcal{S},\mathsf{RO}}(\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda),$$

*where the experiment* POPRF *is defined in Figure 2.*

*Remark 1.* The oracle Prim captures access to the random oracle(s) used in the OPRF construction. We also remark that the original pseudorandomness security game given in [TCR+22] additionally included a LimitEval oracle, which was given to the simulator $\mathcal{S}$ inside the BlindEval and Prim oracles. The LimitEval oracle allowed the simulator to obtain a limited number of evaluation outputs, which is quite useful when proving the security of the underlying OPRF construction using a "one-more"-type assumption. However, we do not use any such assumption, and hence, our simulator does not need the LimitEval oracle. Therefore, we opted to remove it from the security model for the sake of clarity.

Lastly, we define the *request privacy*, which provides security against malicious servers in the preprocessing model. We present the stronger POPRIV2 definition satisfied by our proposal, and refer the reader to [DDT25] for the weaker POPRIV1 definition.

**Definition 4 (Request Privacy).** *We say that a POPRF* F *has request privacy against malicious servers if for all* PPT *adversaries, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that the following holds,*

$$\mathsf{Adv}^{\mathsf{po\text{-}priv\text{-}2}}_{\mathsf{F},\mathcal{A},\mathsf{RO}}(\lambda) = \left|\Pr\left[\mathsf{POPRIV2}^{1}_{\mathsf{F},\mathcal{A},\mathsf{RO}}(\lambda)=1\right] - \Pr\left[\mathsf{POPRIV2}^{0}_{\mathsf{F},\mathcal{A},\mathsf{RO}}(\lambda)=1\right]\right| \leq \mathsf{negl}(\lambda),$$

*where the experiment* POPRIV2 *is defined in Figure 2.*

### 2.2 Lattice Preliminaries

We start with the definitions of standard problems in lattice-based cryptography.

**Definition 5** ($\mathsf{MSIS}_{n,m,\beta}$)**.** *Let* $\mathbf{A} \xleftarrow{\$} R_q^{n\times m}$*, where* $n,\ m > 0$ *and let* $0 < \beta < q$*. We say* $\mathbf{z} \in R_q^m$ *is a solution for* $\mathsf{MSIS}_{n,m,B}$ *problem if* $\mathbf{Az} = 0$ *over* $R_q$ *and* $0 < \|\mathbf{z}\| < \beta$*. If the following inequality holds for an adversary* $\mathcal{A}$

$$\Pr\left[0 < \|\mathbf{z}\| < \beta \wedge \mathbf{Az} = 0 \mid \mathbf{A} \xleftarrow{\$} R_q^{n\times m}, \mathbf{z} \leftarrow \mathcal{A}(\mathbf{A})\right] \geq \epsilon,$$

*then we say* $\mathcal{A}$ *has advantage* $\epsilon$ *in solving* $\mathsf{MSIS}_{n,m,B}$*.*

**Definition 6** ($\mathsf{MLWE}_{\ell,q,m,r,\chi_e,\chi_k}$)**.** *Let* $\chi_e$ *and* $\chi_k$ *be error and secret distributions over* $R$ *respectively,* $\mathbf{A} \xleftarrow{\$} R_q^{\ell\times m}$*, where* $\ell$ *denotes the number of samples,* $m > 0$ *denotes the dimension per secret vector, let* $\mathbf{S} \xleftarrow{\$} \chi_k^{m\times r}$ *be a matrix of* $r$ *secret vectors, and* $\mathbf{E} \xleftarrow{\$} \chi_e^{\ell\times r}$ *be a matrix of* $r$ *error vectors. The* $\mathsf{MLWE}_{\ell,q,m,r,\chi_e,\chi_k}$ *problem asks an adversary* $\mathcal{A}$ *to distinguish between* $(\mathbf{A}, \mathbf{AS}+\mathbf{E})$ *and* $(\mathbf{A},\mathbf{B})$*, for* $\mathbf{B} \xleftarrow{\$} R_q^{\ell\times r}$*. We say* $\mathcal{A}$ *has advantage* $\epsilon$ *against the* $\mathsf{MLWE}_{\ell,q,m,r,\chi_e,\chi_k}$ *problem if*

$$\begin{aligned}&\Pr\left[b=1 \mid \mathbf{A} \xleftarrow{\$} R_q^{\ell\times m}, \mathbf{S} \xleftarrow{\$} \chi_k^{m\times r}, \mathbf{E} \xleftarrow{\$} \chi_e^{\ell\times r}, \mathbf{B} := \mathbf{AS}+\mathbf{E}, b \leftarrow \mathcal{A}(\mathbf{A},\mathbf{B})\right]\\ -&\Pr\left[b=1 \mid \mathbf{A} \xleftarrow{\$} R_q^{\ell\times m}, \mathbf{B} \xleftarrow{\$} R_q^{\ell\times r}, b \leftarrow \mathcal{A}(\mathbf{A},\mathbf{B})\right] \geq \epsilon.\end{aligned}$$

If $r = 1$ in the above MLWE definition, then we simply write $\mathsf{MLWE}_{\ell,q,m,\chi_e,\chi_k}$. We also sometimes drop the parameter $\ell$ when the number of samples is not critical for the discussion.

**Definition 7** ($\mathsf{knMLWE}_{n,m,h,\chi}$)**.** *Let* $\chi$ *be an error distribution over* $R$*,* $\mathbf{A} \xleftarrow{\$} R_q^{n\times m}$*, where* $n > m > 0$*, and let* $\mathbf{S} \xleftarrow{\$} \chi^{h\times n}$*. The* $\mathsf{knMLWE}_{n,m,h,\chi}$ *problem asks an adversary* $\mathcal{A}$ *to distinguish between* $(\mathbf{A}, \mathbf{SA})$ *and* $(\mathbf{A},\mathbf{U})$*, for* $\mathbf{U} \xleftarrow{\$} R_q^{h\times m}$*. We say* $\mathcal{A}$ *has advantage* $\epsilon$ *against the* $\mathsf{knMLWE}_{n,m,h,\chi}$ *problem if*

$$\begin{aligned}&\Pr\left[b=1 \mid \mathbf{A} \xleftarrow{\$} R_q^{n\times m}, \mathbf{S} \xleftarrow{\$} \chi^{h\times n}, b \leftarrow \mathcal{A}(\mathbf{A},\mathbf{SA})\right]\\ -&\Pr\left[b=1 \mid \mathbf{A} \xleftarrow{\$} R_q^{n\times m}, \mathbf{U} \xleftarrow{\$} R_q^{h\times m}, b \leftarrow \mathcal{A}(\mathbf{A},\mathbf{U})\right] \geq \epsilon.\end{aligned}$$

The duality between the above Knapsack form of MLWE and regular MLWE has been established already. That is, from a practical security perspective, $\mathsf{knMLWE}_{n,m,h,\chi}$ is as hard as $\mathsf{MLWE}_{n-m,m,\chi}$, which we will use to estimate the practical hardness of $\mathsf{knMLWE}_{n,m,h,\chi}$. We refer to [EZS+19, Appendix C] for a summary discussion (and references therein for more details). Also, note that our $\mathsf{knMLWE}$ definition is in the "multi-secret" setting where multiple vectors are multiplied by the matrix $\mathbf{A}$. A standard hybrid argument of replacing each $\mathbf{s}_i^\top \mathbf{A}$ with a uniformly random $\mathbf{u}_i^\top$ reduces its hardness to standard Knapsack form of MLWE.

The second parameter $m$ in the definitions of $\mathsf{MSIS}$ and $\mathsf{knMLWE}$ does not play a critical role in our practical hardness estimations of these problems against state-of-the-art attacks (i.e., we assume that the adversary has the optimal choice of $m$). Therefore, we sometimes simply omit this parameter.

**Definition 8 (Discrete Gaussian Distribution).** *For any $s > 0$ and dimension $n \in \mathbb{Z}^+$, the* spherical *$n$-dimensional Gaussian function with parameter $s$[9] is defined as $\rho_s(\mathbf{x}) := \exp\left(-\pi\|\mathbf{x}\|^2/s^2\right)$ for $\mathbf{x} \in \mathbb{R}^n$. More generally, given a positive definite symmetric covariance parameter matrix $\mathbf{\Sigma} \in \mathbb{R}^{n\times n}$ and center $\mathbf{c}$, the $n$-dimensional Gaussian function with covariance parameter $\mathbf{\Sigma}$ and center $\mathbf{c}$ is defined as $\rho_{\Sigma,\mathbf{c}}(\mathbf{x}) := \exp(-\pi(\mathbf{x}-\mathbf{c})^\top \mathbf{\Sigma}^{-1}(\mathbf{x}-\mathbf{c}))$ for $\mathbf{x} \in \mathbb{R}^n$. The discrete Gaussian distribution $\mathcal{D}_{\Lambda,\mathbf{\Sigma},\mathbf{c}}$ over an $n$-dimensional lattice $\Lambda \subseteq \mathbb{R}^n$ with covariance parameter $\mathbf{\Sigma}$, centre $\mathbf{c}$ and support $\Lambda$ is defined as $D_{\Lambda,\mathbf{\Sigma},\mathbf{c}}(\mathbf{x}) := \frac{\rho_{\mathbf{\Sigma},\mathbf{c}}(\mathbf{x})}{\sum_{\mathbf{y}\in\Lambda}\rho_{\mathbf{\Sigma},\mathbf{c}}(\mathbf{y})}$ for $\mathbf{x} \in \Lambda$. In the spherical case, where $\mathbf{\Sigma} = s^2\mathbf{I}_n$, we write $D_{\Lambda,s,\mathbf{c}}$, and we omit $\mathbf{c}$ if it is $\mathbf{0}$.*

**Lemma 1 ([DPS23],[BLP+13, Lemma 2.3]).** *There exists a $\mathsf{PPT}$ algorithm that, given a basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a full rank $n$-dimensional lattice $\Lambda$, a positive definite symmetric matrix $\mathbf{\Sigma}$ and a center $\mathbf{c} \in \mathbb{R}^n$, returns a sample from $\mathcal{D}_{\Lambda,\mathbf{\Sigma},\mathbf{c}}$, assuming the condition $\sqrt{\ln(2n+4)/\pi} \cdot \max_i \|\mathbf{\Sigma}^{-1/2}\mathbf{b}_i\| \le 1$.*

**Lemma 2 ([KLSS23],[Pei10]).** *For $n \in \mathbb{Z}^+, \epsilon \in \mathbb{R}^+$, let $\mathbf{\Sigma}_1, \mathbf{\Sigma}_2$ be positive definite symmetric matrices such that $\mathbf{\Sigma}_3^{-1} := \mathbf{\Sigma}_1^{-1} + \mathbf{\Sigma}_2^{-1}$ satisfies $\sqrt{\mathbf{\Sigma}_3} \ge \eta_\epsilon(\mathbb{Z}^n)$ for $0 < \epsilon < 1/2$. Then for an arbitrary center $\mathbf{c} \in \mathbb{Z}^n$, the distribution*

$$\{\mathbf{x}_1 + \mathbf{x}_2 : \mathbf{x}_1 \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^n,\mathbf{\Sigma}_1},\ \mathbf{x}_2 \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^n,\mathbf{\Sigma}_2,\mathbf{c}}\},$$

*is within statistical distance $\le 2\epsilon$ of $\mathcal{D}_{\mathbb{Z}^n,\mathbf{\Sigma}_1+\mathbf{\Sigma}_2,\mathbf{c}}$.*

**Definition 9 (Smoothing Parameter [MR04, Pei10]).** *For an $n$-dimensional lattice $\Lambda$ and $\epsilon > 0$, the smoothing parameter $\eta_\epsilon(\Lambda)$ is the smallest $s$ such that $\rho_{1/s}(\Lambda^* \setminus \mathbf{0}) \le \epsilon$, where $\Lambda^*$ denotes the dual lattice of $\Lambda$. More generally, for a positive definite symmetric matrix $\mathbf{\Sigma}$, we say that $\sqrt{\mathbf{\Sigma}} \ge \eta_\epsilon(\Lambda)$ if $\eta_\epsilon(\sqrt{\mathbf{\Sigma}}^{-1} \cdot \Lambda) \le 1$.*

**Lemma 3 ([MR04]).** *For $n \in \mathbb{Z}^+, \epsilon \in \mathbb{R}^+$, and $n$ dimensional lattice $\Lambda$, we have*

$$\eta_\epsilon(\Lambda) \le \sqrt{\frac{\ln(2n(1+1/\epsilon))}{\pi}} \cdot \lambda_n(\Lambda), \tag{1}$$

*where $\lambda_n(\Lambda)$ is the smallest radius of an $n$-dimensional ball that contains $n$ linearly independent vectors in $\Lambda$.*

**Lemma 4 ([KLSS23]).** *For $n \in \mathbb{Z}^+, \epsilon \in \mathbb{R}^+$, $n$ dimensional lattice $\Lambda$, and a positive definite symmetric matrix $\mathbf{\Sigma}$, we have $\sqrt{\mathbf{\Sigma}} \ge \eta_\epsilon(\Lambda)$ if $\|\mathbf{\Sigma}^{-1}\| \le \eta_\epsilon(\Lambda)^{-2}$.*

**Lemma 5 (Generalization of [ENP24, Lemma 1] to non-square hint matrices).** *Fix $s_1 \in \mathbb{R}^+$, $n, \ell \in \mathbb{Z}^+$, a positive definite symmetric matrix $\mathbf{\Sigma}_1$, and a matrix $\mathbf{R} \in \mathbb{Z}^{\ell\times n}$. Furthermore, let $\mathbf{\Sigma}_0 := \left(\mathbf{\Sigma}_1^{-1} + \frac{1}{s_1^2}\mathbf{R}^\top\mathbf{R}\right)^{-1}$. Then, the following two probability distributions over $\mathbb{Z}^{n+\ell}$ are equal:*

$$\mathcal{D}_1 := \left\{(\mathbf{e}, \mathbf{h}) :\ \mathbf{e} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^n,\mathbf{\Sigma}_1},\ \mathbf{e}' \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^\ell,s_1},\ \mathbf{h} = \mathbf{R}\mathbf{e} + \mathbf{e}'\right\}$$

$$\mathcal{D}_2 := \left\{(\widetilde{\mathbf{e}}, \mathbf{h}) :\ \mathbf{e} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^n,\mathbf{\Sigma}_1},\ \mathbf{e}' \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^\ell,s_1},\ \mathbf{h} = \mathbf{R}\mathbf{e} + \mathbf{e}',\ \mathbf{c} := \frac{1}{s_1^2}\mathbf{\Sigma}_0\mathbf{R}\mathbf{h},\ \widetilde{\mathbf{e}} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^\ell,\mathbf{\Sigma}_0,\mathbf{c}}\right\}$$

[9] Note that the parameter $s$ is related to the standard deviation $\sigma$ by $s = \sqrt{2\pi} \cdot \sigma$.

*Proof.* Fix any $(\mathbf{v}, \mathbf{w}) \in \mathbb{Z}^n \times \mathbb{Z}^\ell$. For the first distribution we have:

$$\begin{aligned}\mathcal{D}_1(\mathbf{v}, \mathbf{w}) &= \mathcal{D}_{\mathbb{Z}^n, \mathbf{\Sigma}_1}(\mathbf{v}) \cdot \mathcal{D}_{\mathbb{Z}^n, s_1}(\mathbf{w} - \mathbf{R}\mathbf{v}) \\ &\propto \exp\left[-\pi\left(\mathbf{v}^\top \mathbf{\Sigma}_1^{-1}\mathbf{v} - \frac{1}{s_1^2}(\mathbf{w} - \mathbf{R}\mathbf{v})^\top(\mathbf{w} - \mathbf{R}\mathbf{v})\right)\right] \\ &= \exp\left[-\pi\left((\mathbf{v} - \mathbf{c})^\top \mathbf{\Sigma}_0^{-1}(\mathbf{v} - \mathbf{c}) + \frac{1}{s_1^2}\mathbf{w}^\top\mathbf{w} - \mathbf{c}^\top\mathbf{\Sigma}_0^{-1}\mathbf{c}\right)\right],\end{aligned}$$

where $\mathbf{c} := \frac{1}{s_1^2}\mathbf{\Sigma}_0\mathbf{R}^\top\mathbf{w}$. Since $\frac{1}{s_1^2}\mathbf{w}^\top\mathbf{w} - \mathbf{c}^\top\mathbf{\Sigma}_0^{-1}\mathbf{c}$ is a constant that does not depend on $\mathbf{v}$, it follows that the conditional distribution $\Pr_{(\mathbf{e},\mathbf{h}) \xleftarrow{\$} \mathcal{D}_1}[\mathbf{e} = \mathbf{v} | \mathbf{h} = \mathbf{w}] = \mathcal{D}_{\mathbb{Z}^n, \mathbf{\Sigma}_0, \mathbf{c}}(\mathbf{v})$, which is (by construction of $\mathcal{D}_2$) exactly equal to the conditional distribution $\Pr_{(\widetilde{\mathbf{e}},\mathbf{h}) \xleftarrow{\$} \mathcal{D}_2}[\widetilde{\mathbf{e}} = \mathbf{v} | \mathbf{h} = \mathbf{w}]$ in $\mathcal{D}_2$. Since the marginal distribution of $\mathbf{h}$ is also exactly the same in $\mathcal{D}_1$ and $\mathcal{D}_2$, we conclude that $\mathcal{D}_1 = \mathcal{D}_2$. □

## 3 MLWE-PRF-RU: New Lattice Problem with a Mix of MLWE and PRF Samples

We introduce a family of new (interactive) assumptions where MLWE and (randomized) PRF samples are mixed up. We call this family *MLWE-PRF with Re-Use*, MLWE-PRF-RU, and instantiate a member, $\mathsf{MLWE\text{-}PRF\text{-}RU}_{\mathsf{H}}$, by choosing an appropriate "message mapping" H for the PRF samples.

We prove a set of reductions to establish strong evidence for the hardness of MLWE-PRF-RU. A high-level (simplified) summary of the problems and the reductions we achieve is depicted in Figure 1. We first introduce an intermediary problem, Hint-MLWE-PRF, and show a reduction from Hint-MLWE-PRF to MLWE-PRF-RU. Here, we interpret the error information leaked in MLWE-PRF-RU as additional hints (hence the name of the problem). Then, we introduce the main problem that forms the basis for the hardness of MLWE-PRF-RU, namely MLWE-PRF. This problem is a mild variant of the MLWE problem, where the adversary is additionally given randomized PRF outputs. We show a reduction from MLWE-PRF to Hint-MLWE-PRF, which overall leads a reduction from MLWE-PRF to MLWE-PRF-RU. As the final piece of the puzzle, we show a reduction from MLWE to MLWE-PRF for natural instantiations of H, including modelling H as a random oracle and using the mapping from BP14 PRF [BP14].

In the MLWE-PRF-RU problem, the adversary is allowed to *adaptively* query an adversarially-chosen matrix $\mathbf{R}$ to an oracle that returns an MLWE sample. After seeing the MLWE samples, the adversary can then (again adaptively) obtain "deficient" randomized PRF samples of the form $\mathsf{H}(x) \cdot \mathbf{k} + \mathbf{e}_p$ with $\mathbf{e}_p$ dependent on the MLWE error $\mathbf{e}$ and the matrix $\mathbf{R}$, where $\mathbf{e}_p$ has the form $\mathbf{e}_p = \mathbf{e}' - \mathbf{R}\mathbf{e}$, and $\mathbf{e}'$ is a fresh 'masking' error. The assumption states that this real case is indistinguishable from random. Importantly, although the MLWE samples (returned by SampMLWEcom oracle in Figure 3) are independent of $\mathbf{R}$, the adversary has to *commit* to $\mathbf{R}$ in advance of seeing the MLWE samples (thus $\mathbf{R}$ is an input of SampMLWEcom) since the MLWE error is reused in the deficient PRF samples. Looking ahead, this matches the (adversarial) behaviour in the OPRF pseudorandomness proof. We formally define the problem below.

**Definition 10** ($\mathsf{MLWE\text{-}PRF\text{-}RU}_{\mathsf{H},Q,Q_x^\infty,Q_\mathsf{M},q,m,N,h,L,\beta_r,\overrightarrow{\chi}}$). *Let $\overrightarrow{\chi} = (\chi, \chi_1, \chi_k)$ be discrete distributions over $R$,*

*let* H *be a hash family, and let $Q, Q_x^\infty, Q_\mathsf{M}, q, m, N, h, L, \beta_r \in \mathbb{Z}^+$. We say that the MLWE-PRF with Re-Use assumption* $\mathsf{MLWE\text{-}PRF\text{-}RU}_{\mathsf{param}}$ *holds, for* $\mathsf{param} := (\mathsf{H}, Q, Q_x^\infty, Q_\mathsf{M}, q, m, N, h, L, \beta_r, \overrightarrow{\chi})$*, if for all* PPT *adversaries $\mathcal{A}$, there exists a negligible function* $\mathsf{negl}(\cdot)$*, such that the following holds,*

$$\mathsf{Adv}^{\mathsf{MLWE\text{-}PRF\text{-}RU}}_{\mathsf{param}}(\lambda) = \left|\Pr\left[\mathsf{MLWE\text{-}PRF\text{-}RU}^1(\lambda) = 1\right] - \Pr\left[\mathsf{MLWE\text{-}PRF\text{-}RU}^0(\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda),$$

*where the experiment* $\mathsf{MLWE\text{-}PRF\text{-}RU}^b$ *is defined in Figure 3, $Q_x$ denotes the number of* SampPRF-RU *queries for each $x$, $Q := \sum_x Q_x$ denotes the total number of* SampPRF-RU *queries in the experiment, $Q_x^\infty$ is the maximum allowed upper bound on $\max_x\{Q_x\}$, and $Q_\mathsf{M}$ denotes the total number of* SampMLWEcom *queries.*

$\mathsf{MLWE\text{-}PRF\text{-}RU}^{b}_{\mathsf{H},Q,Q_x^\infty,Q_\mathsf{M},q,m,N,h,L,\beta_r,\vec{\chi}}(\lambda)$

$\mathbf{k} \xleftarrow{\$} \chi_k^m$
$\mathcal{Y}, \mathcal{M} := \emptyset$ // Initialize empty tables
$b' \leftarrow \mathcal{A}^{\mathsf{SampMLWEcom},\, \mathsf{SampPRF\text{-}RU}}(1^\lambda)$
**return** $b = b'$

$\mathsf{SampMLWEcom}(\mathbf{R} \in R^{h\times N})$

**if** $\|\mathbf{R}\|_\infty > \beta_r$, **then return** $\perp$
$\mathbf{A} \xleftarrow{\$} R_q^{N\times m}$, $\mathbf{e} \xleftarrow{\$} \chi^N$
**if** $b = 0$ **then** $\mathbf{c} \xleftarrow{\$} R_q^N$
**if** $b = 1$ **then** $\mathbf{c} := \mathbf{A}\mathbf{k} + \mathbf{e} \in R_q^N$
$\mathcal{M}[\mathbf{A}] := (\mathbf{c}, \mathbf{e}, \mathbf{R})$
**return** $(\mathbf{A}, \mathbf{c})$

$\mathsf{SampPRF\text{-}RU}(\mathbf{A},\, x \in \{0,1\}^L)$

**if** $\mathcal{M}[\mathbf{A}] = \perp$, **then return** $\perp$
$(\cdot, \mathbf{e}, \mathbf{R}) := \mathcal{M}[\mathbf{A}]$
$\mathbf{e}' \xleftarrow{\$} \chi_1^h$
$\mathbf{B}_x := \mathsf{H}(x) \in R_q^{h\times m}$
**if** $b = 0$ **then**
  **if** $\mathcal{Y}[x] = \perp$ **then** $\mathcal{Y}[x] \xleftarrow{\$} R_q^h$
  $\mathbf{z} := \mathcal{Y}[x] + \mathbf{e}' - \mathbf{R}\mathbf{e} \in R_q^h$
**if** $b = 1$ **then**
  $\mathbf{z} := \mathbf{B}_x\mathbf{k} + \mathbf{e}' - \mathbf{R}\mathbf{e} \in R_q^h$
$\mathcal{M}[\mathbf{A}] := \perp$
**return** $\mathbf{z}$

Fig. 3: MLWE-PRF-RU security game.

Now, the MLWE-PRF problem is the assumption that the randomized function $x \mapsto \mathsf{H}(x) \cdot \mathbf{k} + \mathbf{e}_B$ (for "small" Gaussian distributed error $\mathbf{e}_B$) is pseudorandom, even given some additional MLWE samples $(\mathbf{A}, \mathbf{A}\mathbf{k} + \mathbf{e})$ with $\mathbf{A}$ having uniformly random entries in $R_q$ and $\mathbf{e}$ are "small" Gaussian errors. Thus, MLWE-PRF is a mild variant of the standard PRF pseudorandomness assumption under the function H, i.e. $x \mapsto \lfloor \mathsf{H}(x) \cdot \mathbf{k} \rceil_p$, which has a security reduction from MLWR (and MLWE) when H is instantiated as in [BLMR13] or [BP14].

**Definition 11** ($\mathsf{MLWE\text{-}PRF}_{\mathsf{H},Q_\mathsf{P},Q_\mathsf{M},q,m,h,L,\vec{\chi}}$)**.** *Let $\vec{\chi} = (\chi_0, \chi_{0,B}, \chi_k)$ be discrete distributions over $R$, let* H *be a hash family, and let $Q_\mathsf{P}, Q_\mathsf{M}, q, m, h, L \in \mathbb{Z}^+$. We say that the* $\mathsf{MLWE\text{-}PRF}_{\mathsf{param}}$ *assumption holds, for* $\mathsf{param} := (\mathsf{H}, Q_\mathsf{P}, Q_\mathsf{M}, q, m, h, L, \vec{\chi})$*, if for all* PPT *adversaries $\mathcal{A}$, there exists a negligible function* $\mathsf{negl}(\cdot)$*, such that the following holds,*

$$\mathsf{Adv}^{\mathsf{MLWE\text{-}PRF}}_{\mathsf{param}}(\lambda) = \left|\Pr\left[\mathsf{MLWE\text{-}PRF}^1(\lambda) = 1\right] - \Pr\left[\mathsf{MLWE\text{-}PRF}^0(\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda),$$

*where the experiment* $\mathsf{MLWE\text{-}PRF}^b$ *is defined in Figure 4, $Q_\mathsf{P}$ denotes the number of queries to the* SampPRF *oracle, while $Q_\mathsf{M}$ denotes the number of queries to the* SampMLWE *oracle.*

We present our reduction from MLWE-PRF to MLWE-PRF-RU via an intermediate problem, which is a "hint" variant of the MLWE-PRF-RU problem. Here, we provide the adversary with (i) MLWE hints of the form $\mathbf{h}_i := \mathbf{R}_i\mathbf{e}_i + \bar{\mathbf{e}}'_i$ for the MLWE error $\mathbf{e}_i$, and (ii) PRF hints of the form $\mathbf{h}_{B,i} := \mathbf{e}_B + \mathbf{e}'_{B,i}$ for the PRF error $\mathbf{e}_B$. We call the latter variant as the Hint-MLWE-PRF problem. Our Hint-MLWE-PRF problem can be viewed as an extension of the "Matrix-Hint-MLWE" problem introduced in [ENP24] (with a reduction from MLWE), which in turn generalizes the Hint-MLWE problem introduced in [KLSS23]. The "Matrix" variant in [ENP24] allows for general hint matrices, rather than block diagonal hint matrices used in [KLSS23]. Extending on these, in our Hint-MLWE-PRF problem, we allow for *adaptively-chosen* general hint matrices for MLWE samples as well as hints for PRF samples that may have *non-random* matrices.

**Definition 12** ($\mathsf{Hint\text{-}MLWE\text{-}PRF}_{\mathsf{H},Q,Q_x^\infty,Q_\mathsf{M},q,m,N,h,L,\beta_r,\vec{\chi}}$)**.** *Let $\vec{\chi} = (\chi, \chi_B, \bar{\chi}_1, \chi_{1,B}, \chi_k)$ be discrete distributions over $R$, let* H *be a hash family, and let $Q, Q_x^\infty, Q_\mathsf{M}, q, m, N, h, L, \beta_r \in \mathbb{Z}^+$. We say that the* $\mathsf{Hint\text{-}MLWE\text{-}PRF}_{\mathsf{param}}$ *holds, for* $\mathsf{param} := (\mathsf{H}, Q, Q_x^\infty, Q_\mathsf{M}, q, m, N, h, L, \beta_r, \vec{\chi})$*, if for all* PPT *adversaries $\mathcal{A}$, there exists a negligible function* $\mathsf{negl}(\cdot)$*, such that the following holds,*

$$\left|\Pr\left[\mathsf{Hint\text{-}MLWE\text{-}PRF}^1(\lambda) = 1\right] - \Pr\left[\mathsf{Hint\text{-}MLWE\text{-}PRF}^0(\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda),$$

$\mathsf{MLWE\text{-}PRF}^b_{\mathsf{H},Q_\mathsf{P},Q_\mathsf{M},q,m,h,L,\vec{\chi}}(\lambda)$

$\mathbf{k} \xleftarrow{\$} \chi_k^m$
$\mathcal{Y} := \emptyset$ // Initialize empty table
$b' \leftarrow \mathcal{A}^{\mathsf{SampPRF},\mathsf{SampMLWE}}(1^\lambda)$
**return** $b = b'$

$\mathsf{SampMLWE}()$

$\mathbf{a} \xleftarrow{\$} R_q^m, e \xleftarrow{\$} \chi_0$
**if** $b = 0$, **then** $c \xleftarrow{\$} R_q$
**if** $b = 1$, **then** $c := \mathbf{a}^\top \mathbf{k} + e \in R_q$
**return** $(\mathbf{a}, c)$

$\mathsf{SampPRF}(x)$

**if** $\mathcal{Y}[x] = \perp$ **then**
  **if** $b = 0$ **then**
    $\mathcal{Y}[x] \xleftarrow{\$} R_q^h$
  **if** $b = 1$ **then**
    $\mathbf{B}_x := \mathsf{H}(x) \in R_q^{h \times m}$
    $\mathbf{e}_B \xleftarrow{\$} \chi_{0,B}^h$
    $\mathcal{Y}[x] := \mathbf{B}_x \mathbf{k} + \mathbf{e}_B \in R_q^h$
**return** $\mathcal{Y}[x]$

Fig. 4: MLWE-PRF security game.

*where the experiment* Hint-MLWE-PRF$^b$ *is defined in Figure 5,* $Q_x$ *denotes the number of* SampHintPRF *queries for each* $x$, $Q := \sum_x Q_x$ *denotes the total number of* SampHintPRF *queries in the experiment,* $Q_x^\infty$ *is the maximum allowed upper bound on* $\max_x\{Q_x\}$, *and* $Q_\mathsf{M}$ *denotes the total number of* SampHintMLWE *queries.*

Observe that in the Hint-MLWE-PRF experiment in Figure 5, once a particular $x$ value is queried to SampHintPRF for the first time, the entries $\mathcal{Y}[x]$ and $\mathcal{E}[x]$ are fixed. So for any subsequent queries with the same $x$, the only elements that change are the masking error $\mathbf{e}'_B$ and the corresponding hint $\mathbf{h}_B$. This behaviour matches with the computation of the same $\mathbf{B}_x\mathbf{k}$ term (or $\mathcal{Y}[x]$) in the MLWE-PRF-RU experiment in Figure 3.

### 3.1 Hardness Reduction from Hint-MLWE-PRF to MLWE-PRF-RU

We start with the first step of our reduction series. The idea here is that in the Hint-MLWE-PRF problem, in addition to the MLWE-PRF samples $(\mathbf{B}_x, \mathbf{y}_B := \mathbf{B}_x\mathbf{k} + \mathbf{e}_B)$ and $(\mathbf{A}, \mathbf{c} := \mathbf{A}\mathbf{k} + \mathbf{e})$, the adversary is also provided with appropriate hint vector $\mathbf{h}_j$ about error $\mathbf{e}$ with respect to the $\mathbf{R}$ matrix provided by the adversary, i.e. $\mathbf{h} := \bar{\mathbf{e}}' - \mathbf{R}\mathbf{e}$, and also another hint $\mathbf{h}_B := \mathbf{e}'_B - \mathbf{e}_B$. The hints $\mathbf{h}$ and $\mathbf{h}_B$ then allow the adversary to simulate the "mixed" MLWE-PRF-RU samples of the form $(\mathbf{c} := \mathbf{A}\mathbf{k} + \mathbf{e}, \mathbf{z} := \mathbf{B}_x\mathbf{k} + \hat{\mathbf{e}})$ from the given MLWE-PRF samples, since $\mathbf{y}_B + \mathbf{h} + \mathbf{h}_B = \mathbf{B}_x\mathbf{k} + \mathbf{e}'_B + \bar{\mathbf{e}}' - \mathbf{R}\mathbf{e}$, where $\hat{\mathbf{e}} := \mathbf{e}'_B + \bar{\mathbf{e}}' - \mathbf{R}\mathbf{e}$ is the simulated error term for $\mathbf{z}$. By the discrete Gaussian convolution Lemma 2, the term $\mathbf{e}'_B + \bar{\mathbf{e}}'$ is statistically close to a discrete Gaussian with parameter $s_1 := \sqrt{s_{1,B}^2 + \bar{s}_1^2}$ when the independent errors $\mathbf{e}'_B$ and $\bar{\mathbf{e}}'$ are discrete Gaussians with parameters $s_{1,B}$ and $\bar{s}_1$, respectively.

**Theorem 1.** *Let* $\vec{\chi} := (\chi, \chi_B, \chi_1, \chi_k)$ *and* $\vec{\chi}_h := (\chi, \chi_B, \bar{\chi}_1, \chi_{1,B}, \chi_k)$ *be discrete distributions over* $R$, *let* $\mathsf{H}$ *be a hash family, and let* $Q, Q_x^\infty, Q_\mathsf{M}, q, m, N, h, L, \beta_r \in \mathbb{Z}^+, \epsilon \in \mathbb{R}^+$. *The* $\mathsf{MLWE\text{-}PRF\text{-}RU}_{\mathsf{H},Q,Q_x^\infty,Q_\mathsf{M},q,m,N,h,L,\beta_r,\vec{\chi}}$ *assumption holds if the* $\mathsf{Hint\text{-}MLWE\text{-}PRF}_{\mathsf{H},Q,Q_x^\infty,Q_\mathsf{M},q,m,N,h,L,\beta_r,\vec{\chi}_h}$ *assumption holds. More precisely, for any* PPT *adversary* $\mathcal{A}$ *against the* MLWE-PRF-RU *problem, there exists a* PPT *adversary* $\mathcal{B}$ *against the* Hint-MLWE-PRF *problem, such that*

$$\mathsf{Adv}^{\mathsf{MLWE\text{-}PRF\text{-}RU}}_{\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{Hint\text{-}MLWE\text{-}PRF}}_{\mathcal{B}}(\lambda) + \bar{\epsilon},$$

*where* $\bar{\epsilon} = 0$ *if* $\chi_1 := \chi_{1,B} + \bar{\chi}_1$ *is the convolution of the distributions* $\chi_{1,B}$ *and* $\bar{\chi}_1$. *Moreover, if* $\chi_1 := \mathcal{D}_{\mathbb{Z}^d,s_1}$, $\chi_{1,B} := \mathcal{D}_{\mathbb{Z}^d,s_{1,B}}$ *and* $\bar{\chi}_1 := \mathcal{D}_{\mathbb{Z}^d,\bar{s}_1}$ *with* $s_1 := \sqrt{s_{1,B}^2 + \bar{s}_1^2}$, *and* $s_1 \geq \sqrt{\ln(2hd(1+1/\epsilon))/\pi}$, *then the above advantage bound holds with* $\bar{\epsilon} = 2Q \cdot \epsilon$.

$\mathsf{Hint\text{-}MLWE\text{-}PRF}^{b}_{\mathsf{H},Q,Q_x^\infty,Q_\mathsf{M},q,m,N,h,L,\beta_r,\vec{\chi}}(\lambda)$

$\mathbf{k} \xleftarrow{\$} \chi_k^m$
$\mathcal{Y}, \mathcal{E}, \mathcal{M} := \emptyset$ // Initialize empty tables
$b' \leftarrow \mathcal{A}^{\mathsf{SampHintMLWE},\,\mathsf{SampHintPRF}}(1^\lambda)$
**return** $b = b'$

$\mathsf{SampHintMLWE}(\mathbf{R})$ // $\mathbf{R} \in R^{h\times N}$

**if** $\|\mathbf{R}\|_\infty > \beta_r$, **then return** $\perp$
$\mathbf{A} \xleftarrow{\$} R_q^{N\times m}$, $\mathbf{e} \xleftarrow{\$} \chi^N$ // MLWE matrix/error
$\bar{\mathbf{e}}' \xleftarrow{\$} \bar{\chi}_1^h$ // MLWE hint masking term
$\mathbf{h} := \bar{\mathbf{e}}' - \mathbf{R}\mathbf{e} \in R^h$
**if** $b = 0$, **then** $\mathbf{c} \xleftarrow{\$} R_q^N$
**if** $b = 1$, **then** $\mathbf{c} := \mathbf{A}\mathbf{k} + \mathbf{e} \in R_q^N$
**return** $(\mathbf{A}, \mathbf{c}, \mathbf{h})$

$\mathsf{SampHintPRF}(x)$ // $x \in \{0,1\}^L$

**if** $\mathcal{Y}[x] = \perp$ **then**
  $\mathcal{E}[x] \xleftarrow{\$} \chi_B^h$
  **if** $b = 0$ **then**
    $\mathcal{Y}[x] \xleftarrow{\$} R_q^h$
  **if** $b = 1$ **then**
    $\mathbf{B}_x := \mathsf{H}(x) \in R_q^{h\times m}$
    $\mathcal{Y}[x] := \mathbf{B}_x\mathbf{k} + \mathcal{E}[x] \in R_q^h$
$\mathbf{e}'_B \xleftarrow{\$} \chi_{1,B}^h$ // PRF hint masking
$\mathbf{h}_B := \mathbf{e}'_B - \mathcal{E}[x]$
**return** $(\mathbf{B}_x, \mathcal{Y}[x], \mathbf{h}_B)$

Fig. 5: Hint-MLWE-PRF security game.

*Proof.* Given an adversary $\mathcal{A}$ against MLWE-PRF-RU, we construct an adversary $\mathcal{B}$ against Hint-MLWE-PRF that runs as shown in Figure 6. Note that to be able to get a meaningful response from SimPRF-RU, the adversary $\mathcal{A}$ has to query SimMLWEcom first and use a matrix $\mathbf{A}$ returned from SimMLWEcom to query SimPRF-RU. This is exactly the same as in the original game MLWE-PRF-RU (with the corresponding oracles SampMLWEcom and SampPRF-RU). Also, the adversary $\mathcal{A}$ is allowed to query SimPRF-RU once per $\mathbf{A}$, again matching the original MLWE-PRF-RU game. Therefore, without loss of generality, we assume that $\mathcal{A}$ queries SimPRF-RU exactly once for a matrix $\mathbf{A}$ and always uses a matrix $\mathbf{A}$ returned by SimMLWEcom.

$\mathcal{B}^{\mathsf{SampHintMLWE},\mathsf{SampHintPRF}}_{\mathsf{H},Q,Q_x^\infty,q,m,N,h,L,\beta_r,\vec{\chi}_h}(1^\lambda)$

$\mathcal{H}_a := \emptyset$
$b' \leftarrow \mathcal{A}^{\mathsf{SimMLWEcom},\,\mathsf{SimPRF\text{-}RU}}(1^\lambda)$
**return** $b = b'$

$\mathsf{SimMLWEcom}(\mathbf{R})$

$(\mathbf{A}, \mathbf{c}, \mathbf{h}) \leftarrow \mathsf{SampHintMLWE}(\mathbf{R})$
$\mathcal{H}_a[\mathbf{A}] := \mathbf{h}$ // store MLWE hint
**return** $(\mathbf{A}, \mathbf{c})$

$\mathsf{SimPRF\text{-}RU}(\mathbf{A}, x)$

**if** $\mathcal{H}_a[\mathbf{A}] = \perp$, **then return** $\perp$
$(\mathbf{B}_x, \mathbf{y}_B, \mathbf{h}_B) \leftarrow \mathsf{SampHintPRF}(x)$
$\mathbf{h} := \mathcal{H}_a[\mathbf{A}]$ // retrieve MLWE hint
$\mathbf{z} := \mathbf{y}_B + \mathbf{h}_B + \mathbf{h}$
$\mathcal{H}_a[\mathbf{A}] = \perp$
**return** $\mathbf{z}$

Fig. 6: Hint-MLWE-PRF solver algorithm $\mathcal{B}$ given an MLWE-PRF-RU solver algorithm $\mathcal{A}$.

Next, to analyze the advantage of $\mathcal{B}$ against Hint-MLWE-PRF, we consider first the case that $b = 1$. In this case, for a query $\mathbf{R}$ to SimMLWEcom, followed by a query $(\mathbf{A}, x)$ to SimPRF-RU, the overall response $(\mathbf{A}, \mathbf{c}, \mathbf{z})$ returned has $\mathbf{c} = \mathbf{A}\mathbf{k} + \mathbf{e}$, and $\mathbf{z} = \mathbf{y}_B + \mathbf{h}_B + \mathbf{h}$ with $\mathbf{y}_B = \mathbf{B}_x\mathbf{k} + \mathbf{e}_B$, $\mathbf{B}_x = \mathsf{H}(x)$, $\mathbf{h}_B = \mathbf{e}'_B - \mathbf{e}_B$, and $\mathbf{h} = \bar{\mathbf{e}}' - \mathbf{R}\mathbf{e}$, where $\mathbf{e}_B$ and $\mathbf{e}'_B$ are sampled from the distributions $\chi_B^h$ and $\chi_{1,B}^h$, respectively. Hence, we

have $\mathbf{z} = \mathbf{B}_x\mathbf{k} + \hat{\mathbf{e}}$ for $\hat{\mathbf{e}} := \mathbf{e}'_B + \bar{\mathbf{e}}' - \mathbf{R}\mathbf{e}$. Therefore, in the Hint-MLWE-PRF$^1$ game, the view of $\mathcal{A}$ is simulated with exactly the same distribution as in the real MLWE-PRF-RU$^1$ game in the case $\chi_1 = \chi_{1,B} + \bar{\chi}_1$.

In the other case that $b = 0$, for a similar query pair $(\mathbf{R}, x)$ of $\mathcal{A}$, the overall response $(\mathbf{A}, \mathbf{c}, \mathbf{z})$ returned has uniformly random $\mathbf{c}$ in $R_q^N$, and $\mathbf{z} = \mathbf{y}_B + \mathbf{h}_B + \mathbf{h} = (\mathcal{Y}[x] - \mathcal{E}[x]) + (\mathbf{e}'_B + \bar{\mathbf{e}}') - \mathbf{R}\mathbf{e}$. On the other hand, in the MLWE-PRF-RU$^0$ game, $\mathbf{c}$ is also uniform in $R_q^N$ and $\mathbf{z} = \mathcal{Y}^{RU}[x] + \mathbf{e}' - \mathbf{R}\mathbf{e}$, where $\mathcal{Y}^{RU}[x]$ denotes the $\mathcal{Y}$ oracle in the MLWE-PRF-RU$^0$ game. Since $\mathcal{Y}^{RU}[x]$ and $(\mathcal{Y}[x] - \mathcal{E}[x])$ are both uniformly random in $R_q^h$ and independent for distinct $x$ in both games, $(\mathbf{e}'_B + \bar{\mathbf{e}}')$ and $\mathbf{e}'$ are both distributed as $\chi_1^h$ in both games if $\chi_{1,B} + \bar{\chi}_1 = \chi_1$. Therefore, in the Hint-MLWE-PRF$^0$ game, the view of $\mathcal{A}$ is simulated with exactly the same distribution as in the MLWE-PRF-RU$^0$ game in the case $\chi_1 = \chi_{1,B} + \bar{\chi}_1$.

We conclude that the advantage of $\mathcal{B}$ against Hint-MLWE-PRF is equal to the advantage of $\mathcal{A}$ against MLWE-PRF-RU, as claimed, in the case $\chi_1 = \chi_{1,B} + \bar{\chi}_1$.

The last part of the Theorem follows from the convolution Lemma 2 that implies that $\mathcal{D}_{\mathbb{Z}^{hd}, s_{1,B}} + \mathcal{D}_{\mathbb{Z}^{hd}, \bar{s}_1}$ is within statistical distance $2\epsilon$ of $\mathcal{D}_{\mathbb{Z}^{hd}, s_1}$ with $s_1 := \sqrt{s_{1,B}^2 + \bar{s}_1^2}$, assuming the smoothing condition $s_1 \geq \sqrt{\ln(2hd(1+1/\epsilon))/\pi}$ is satisfied. So for $Q$ queries the statistical distance is at most $2Q \cdot \epsilon$, as claimed. □

### 3.2 Hardness Reduction from MLWE-PRF to Hint-MLWE-PRF

Our reduction from MLWE-PRF to Hint-MLWE-PRF can be viewed as an extension of the reduction for the Matrix-Hint-MLWE problem in [ENP24] from the MLWE problem. However, unlike the Matrix-Hint-MLWE problem in [ENP24], which only involves MLWE samples with respect to *random* matrices, our Hint-MLWE-PRF problem also provides the adversary with MLWE-like samples with respect to possibly *non-random* matrices $\mathbf{B}_x := \mathsf{H}(x)$ for the chosen instantiation of $\mathsf{H}$ (which is not necessarily a random oracle). That is the reason why we need the additional PRF oracle in the MLWE-PRF problem. Therefore, our reduction to Hint-MLWE-PRF is from MLWE-PRF, rather than a reduction from MLWE as in [ENP24]. Our reduction also handles *adaptive* hint-matrix queries unlike prior work [KLSS23, ENP24].

**Theorem 2.** *Let $\overrightarrow{\chi}_h := (\chi, \chi_B, \bar{\chi}_1, \chi_{1,B}, \chi_k)$ be discrete distributions over $R$, where $\chi := \mathcal{D}_{\mathbb{Z}^d, s}$, $\chi_B := \mathcal{D}_{\mathbb{Z}^d, s_B}$, $\bar{\chi}_1 := \mathcal{D}_{\mathbb{Z}^d, \bar{s}_1}$ and $\chi_{1,B} := \mathcal{D}_{\mathbb{Z}^d, s_{1,B}}$ for some $s, s_B, \bar{s}_1, s_{1,B}, \eta, \delta, \epsilon, \epsilon_p \in \mathbb{R}^+$. Let $\mathsf{H}$ be a hash family, and let $Q, Q_x^\infty, Q_\mathsf{M}, q, m, N, h, L, \beta_r \in \mathbb{Z}^+$. Moreover, let $\overrightarrow{\chi} := (\chi_0, \chi_{0,B}, \chi_k)$, where $\chi_0 := \mathcal{D}_{\mathbb{Z}^d, s_0}$ and $\chi_{0,B} := \mathcal{D}_{\mathbb{Z}^d, \hat{\mathbf{\Sigma}}_{0,B}}$, where $s_{0,B} := \sqrt{\|\hat{\mathbf{\Sigma}}_{0,B}\|}$ and $\hat{\rho}_{0,B} := \sigma_{\max}(\hat{\mathbf{\Sigma}}_{0,B})/\sigma_{\min}(\hat{\mathbf{\Sigma}}_{0,B}) = s_{0,B}^2/\sigma_{\min}(\hat{\mathbf{\Sigma}}_{0,B})$. The Hint-MLWE-PRF$_{\mathsf{H},Q,Q_x^\infty,Q_\mathsf{M},q,m,N,h,L,\beta_r,\overrightarrow{\chi}_h}$ assumption holds if the MLWE-PRF$_{\mathsf{H},Q,N\cdot Q_\mathsf{M},q,m,h,L,\overrightarrow{\chi}}$ assumption holds and the following are satisfied:*

$$\bar{s}_1/s \geq \eta \cdot \beta_r d\sqrt{hN}, \quad s_{1,B}/s_B \geq \eta \cdot \sqrt{Q_x^\infty} \tag{2}$$

$$\max(s_{0,B}/s_B,\, s_0/s) \leq \sqrt{\frac{1 - 1/(1+\eta^2) - \epsilon_p}{(1+\delta)}}, \tag{3}$$

$$s_0 \geq \sqrt{\frac{(1+1/\delta)\cdot\ln(2Nd(1+1/\epsilon))}{\pi}}, \quad s_{0,B} \geq \sqrt{\frac{(\hat{\rho}_{0,B}+1/\delta)\cdot\ln(2hd(1+1/\epsilon))}{\pi}}, \tag{4}$$

$$s \geq \sqrt{\frac{\ln(2Nd+4)}{\pi\epsilon_p}}, \quad s_B \geq \sqrt{\frac{\ln(2hd+4)}{\pi\epsilon_p}} \tag{5}$$

*More precisely, for any PPT adversary $\mathcal{A}$ against the Hint-MLWE-PRF problem, there exists a PPT adversary $\mathcal{B}$ against the MLWE-PRF problem, such that*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hint\text{-}MLWE\text{-}PRF}}(\lambda) \leq \mathsf{Adv}_{\mathcal{B}}^{\mathsf{MLWE\text{-}PRF}}(\lambda) + 2(Q + Q_\mathsf{M}) \cdot \epsilon,$$

*where $Q$ and $Q_\mathsf{M}$ denote the total number of* SampHintPRF *and* SampHintMLWE *queries, respectively, that the adversary $\mathcal{A}$ makes, and $Q_x^\infty$ is the maximum number of hints requested per $x$ query.*

In the proof of the above theorem, we extend on the approach of the MLWE to Hint-MLWE reduction in [KLSS23] and in particular its generalization in [ENP24] to hints that contain linear combinations of error coordinates, as is the case in our hint matrices. Namely, the security reduction simulates the hints $\mathbf{h}_j$ and $\mathbf{h}_{B,j}$ with their correct marginal distribution and then simulates samples from the conditional distribution of the error vectors $\mathbf{e}_j$ and $\mathbf{e}_B$ respectively given the hints. Here, to deal with adaptive hint matrix $\mathbf{R}_j$ oracle queries, we use the fact that hint $\mathbf{h}_j$ depends only on $\mathbf{R}_j$ and $\mathbf{e}_j$ but not on $\mathbf{e}_i$ for $i \neq j$ and $\mathbf{e}_B$, and the independence of $\mathbf{e}_j$'s and $\mathbf{e}_B$. Hence the conditional distribution of $\mathbf{e}_j$ given the hints depends only on $\mathbf{h}_j$ and $\mathbf{R}_j$ and not on $\mathbf{h}_i$ and $\mathbf{R}_i$ for $i > j$. This allows our reduction to sample the simulated conditional distribution for $\mathbf{e}_j$ adaptively for each queried $\mathbf{R}_j$ as a discrete Gaussian with parameters that depends only on $\mathbf{R}_j$ and $\mathbf{h}_j$, as in the standard Hint-MLWE reductions [KLSS23, ENP24]. For the hints $\mathbf{h}_{B,j}$ on the PRF error $\mathbf{e}_B$, the conditional distribution of $\mathbf{e}_B$ conditioned on the hints does depend on all the $\mathbf{e}_B$ hints $\mathbf{h}_{B,j}$ for that $x$ value over $j \in [Q_x^\infty]$. However, the latter dependence does not cause a problem with adaptive queries, because the $\mathbf{h}_{B,j}$ hints are all with respect to a fixed identity hint matrix, and does not depend on the adaptively queried $\mathbf{R}_j$'s; therefore our security reduction can sample all the simulated $\mathbf{e}_B$ hints $\{\mathbf{h}_{B,j}\}_{j\in[Q_x^\infty]}$ for a given $x$ value in advance when the $x$ value is first queried to the PRF oracle. We remark that it is to handle this adaptive query issue that we reduce to a hint problem with separate hints $\mathbf{h}_j$ and $\mathbf{h}_{B,j}$ rather than a combined hint $\mathbf{h}_j + \mathbf{h}_{B,j}$, as mentioned in the technical overview. Our reduction results in a lower bound on the ratio $s_{1,B}/s \approx \sqrt{Q_x^\infty}$ that is nearly optimal for large $Q_x$, as it approximately matches the minimum value of $s_{1,B}/s$ required to thwart the known "averaging" attack (see Appendix B).

We are now ready to prove our security reduction from MLWE-PRF to Hint-MLWE-PRF.

*Proof (of Theorem 2).* Given an adversary $\mathcal{A}$ against Hint-MLWE-PRF, we construct an adversary $\mathcal{B}$ against MLWE-PRF that runs as shown in Figure 7.

**Proof of sampleability conditions.** Note that for the discrete Gaussian sampling in Line 9 of SimHintMLWE to be PPT sampleable, it is sufficient by Lemma 1 that $\mathbf{\Sigma}$ is positive definite, i.e. $\sigma_{\min}(\mathbf{\Sigma}) > 0$, where $\mathbf{\Sigma} := \mathbf{\Sigma}_0 - s_0^2\mathbf{I}$, and that[10] $\delta_p \cdot M_{\mathbf{\Sigma}} \leq 1$, where $\delta_p := \sqrt{\frac{\ln(2Nd+4)}{\pi}}$ and $M_{\mathbf{\Sigma}}$ denotes the max column norm of $\sqrt{\mathbf{\Sigma}^{-1}}$, which is bounded as $M_{\mathbf{\Sigma}} \leq \sqrt{\sigma_{\max}(\mathbf{\Sigma}^{-1})} \leq \frac{1}{\sqrt{\sigma_{\min}(\mathbf{\Sigma})}} = \frac{1}{\sqrt{\sigma_{\min}(\mathbf{\Sigma}_0)-s_0^2}}$. So the sufficient sampleability condition is

$$\sigma_{\min}(\mathbf{\Sigma}_0) \geq (1+\delta)\cdot s_0^2 + \delta_p^2, \tag{6}$$

for some $\delta \geq 0$. On the other hand, we have: $\sigma_{\min}(\mathbf{\Sigma}_0) = \frac{1}{\|\mathbf{\Sigma}_0^{-1}\|}$ and

$$\begin{aligned}
\|\mathbf{\Sigma}_0^{-1}\| &= \|\mathbf{\Sigma}_1^{-1} + \frac{1}{\bar{s}_1^2}\mathbf{R}^\top\mathbf{R}\| && \text{(by definition)}\\
&\leq \|\mathbf{\Sigma}_1^{-1}\| + \frac{1}{\bar{s}_1^2}\|\mathbf{R}^\top\mathbf{R}\| && \text{(by triangle inequality)}\\
&\leq \frac{1}{s^2} + \frac{1}{\bar{s}_1^2}\|\mathbf{R}\|^2 && \text{(using } \|\mathbf{R}^\top\mathbf{R}\| = \|\mathbf{R}\|^2\text{)}\\
&\leq \frac{1}{s^2} + \frac{1}{\bar{s}_1^2}((\beta_r d)^2 hN) && \text{(using } \|\mathbf{R}\|_\infty \leq \beta_r\text{)}\\
&\leq \frac{1}{s^2}\cdot(1+1/\eta^2). && (7)
\end{aligned}$$

The last inequality uses the assumed lower bound (2) on $\bar{s}_1/s$. Therefore, using (7), we see that the sampleability condition (6) is implied by the desired lower bound $\frac{s^2}{1+1/\eta^2} \geq (1+\delta)s_0^2 + \delta_p^2$, whereas the assumed condition (3) on $s_0/s$ is equivalent to the lower bound $\frac{s^2}{1+1/\eta^2} \geq (1+\delta)s_0^2 + \epsilon_p s^2$. Comparing these assumed

[10] We remark that there is a minor omission in the Hint-MLWE security reduction analysis in [KLSS23, Thm 1] which omitted this extra PPT sampleability condition and only required positive definiteness; it leads to the extra term $\epsilon_p$ in the bound.

$\mathcal{B}^{\mathsf{SampMLWE},\mathsf{SampPRF}}_{\mathsf{H},Q,Q_x^\infty,q,m,N,h,L,\beta_r,\vec{\chi}}(1^\lambda)$

1 : $\mathcal{Y}_{\mathsf{sim}}, \mathcal{H}_B, \mathcal{J}, \mathcal{M} := \emptyset$ // initialize empty tables
2 : $b' \leftarrow \mathcal{A}^{\mathsf{SimHintMLWE},\ \mathsf{SimHintPRF}}(1^\lambda)$
3 : **return** $b'$

$\mathsf{SimHintMLWE}(\mathbf{R})$

1 : **if** $\|\mathbf{R}\|_\infty > \beta_r$, **then return** $\perp$
2 : $\widetilde{\mathbf{e}}' \xleftarrow{\$} \bar{\chi}_1^h$, $\widetilde{\mathbf{e}} \xleftarrow{\$} \chi^N$ // simulated errors in hint
3 : $\mathbf{h} := \widetilde{\mathbf{e}}' - \mathbf{R}\widetilde{\mathbf{e}}$
4 : $(\mathbf{A}, \mathbf{c}') \leftarrow \mathsf{SampMLWE}^N()$ // get $N$ MLWE samples
// next steps: correct the error in $\mathbf{c}'$
5 : $\mathbf{\Sigma}_1 := s^2\mathbf{I} \in \mathbb{Z}^{Nd\times Nd}$
6 : $\mathbf{\Sigma}_0 := \left(\mathbf{\Sigma}_1^{-1} + \frac{1}{\bar{s}_1^2}\mathbf{R}^\top\mathbf{R}\right)^{-1}$
7 : $\mathbf{d}_j := \frac{1}{\bar{s}_1^2}\mathbf{\Sigma}_0\mathbf{R}^\top\mathbf{h}$
8 : $\mathbf{\Sigma} := \mathbf{\Sigma}_0 - s_0^2\mathbf{I}$
9 : $\mathbf{t} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^{Nd},\mathbf{\Sigma},\mathbf{d}}$ // correction term
10 : $\mathbf{c} := \mathbf{c}' + \mathbf{t} \in R_q^N$
11 : **return** $(\mathbf{A}, \mathbf{c}, \mathbf{h})$

$\mathsf{SimHintPRF}(x)$

1 : **if** $\mathcal{J}[x] = \perp$, **then** $\mathcal{J}[x] = 0$
2 : $\mathcal{J}[x] := \mathcal{J}[x] + 1$, $j := \mathcal{J}[x]$
3 : $\mathbf{B}_x := \mathsf{H}(x)$
4 : **if** $\mathcal{Y}_{\mathsf{sim}}[x] = \perp$ **then**
5 : $\quad\widetilde{\mathbf{e}}_B \xleftarrow{\$} \chi_B^h$ // simulated error
6 : $\quad$**for** $i \in [Q_x^\infty]$ **do** // set all hints for given $x$
7 : $\quad\quad\mathbf{e}'_{B,,i} \xleftarrow{\$} \chi_{1,B}^h$
8 : $\quad\quad\mathbf{h}_{B,i} := \mathbf{e}'_{B,i} - \widetilde{\mathbf{e}}_B$
9 : $\quad\quad\mathcal{H}_B[x,i] := \mathbf{h}_{B,i}$
10 : $\quad$**endfor**
11 : $\quad\mathbf{R}_B^\top := -[\mathbf{I}, \cdots, \mathbf{I}] \in \mathbb{Z}^{hd\times Q_x^\infty hd}$
12 : $\quad\mathbf{h}_B^\top := [\mathbf{h}_{B,1}^\top, \ldots, \mathbf{h}_{B,Q_x^\infty}^\top]$
13 : $\quad\mathbf{y}'_B \leftarrow \mathsf{SampPRF}(x)$ // get PRF sample
// next steps: correct the error in $\mathbf{y}'_B$
14 : $\quad\mathbf{\Sigma}_{1,B} := s_B^2\mathbf{I} \in \mathbb{Z}^{hd\times hd}$
15 : $\quad\mathbf{\Sigma}_{0,B} := \left(\mathbf{\Sigma}_{1,B}^{-1} + \frac{1}{s_{1,B}^2}\mathbf{R}_B^\top\mathbf{R}_B\right)^{-1}$
16 : $\quad\mathbf{d}_B := \frac{1}{s_{1,B}^2}\mathbf{\Sigma}_{0,B}\mathbf{R}_B^\top\mathbf{h}_B$
17 : $\quad\mathbf{t}_B \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^{hd},\mathbf{\Sigma}_{0,B}-\hat{\mathbf{\Sigma}}_{0,B},\mathbf{d}_B}$ // correction term
18 : $\quad\mathcal{Y}_{\mathsf{sim}}[x] := \mathbf{y}'_B + \mathbf{t}_B \in R_q^h$
19 : **return** $(\mathbf{B}_x, \mathcal{Y}_{\mathsf{sim}}[x], \mathcal{H}_B[x,j])$

Fig. 7: MLWE-PRF solver algorithm $\mathcal{B}$ given a Hint-MLWE-PRF solver algorithm $\mathcal{A}$.

and desired lower bounds on $\frac{s^2}{1+1/\eta^2}$, we see that the former implies the latter if $\epsilon_p s^2 \geq \delta_p^2$, which in turn is equivalent to the assumed lower bound (5) on $s$, as required.

Similarly, for the discrete Gaussian sampling in Line 17 of SimHintPRF to be PPT sampleable, it is sufficient by Lemma 1 that $\mathbf{\Sigma}_B$ is positive definite, i.e. $\sigma_{\min}(\mathbf{\Sigma}_B) > 0$, where $\mathbf{\Sigma}_B := \mathbf{\Sigma}_{0,B} - \hat{\mathbf{\Sigma}}_{0,B}$ and $\delta_{pB} \cdot M_{\mathbf{\Sigma}_B} \leq 1$, where $\delta_{pB} := \sqrt{\frac{\ln(2hd+4)}{\pi}}$ and $M_{\mathbf{\Sigma}_B}$ denotes the max column norm of $\sqrt{\mathbf{\Sigma}_B^{-1}}$, which is bounded as $M_{\mathbf{\Sigma}_B} \leq \sqrt{\sigma_{\max}(\mathbf{\Sigma}_B^{-1})} \leq \frac{1}{\sqrt{\sigma_{\min}(\mathbf{\Sigma}_B)}} \leq \frac{1}{\sqrt{\sigma_{\min}(\mathbf{\Sigma}_{0,B})-\sigma_{\max}(\hat{\mathbf{\Sigma}}_{0,B})}} = \frac{1}{\sqrt{\sigma_{\min}(\mathbf{\Sigma}_{0,B})-s_{0,B}^2}}$. So the sufficient sampleability condition is

$$\sigma_{\min}(\mathbf{\Sigma}_{0,B}) \geq (1+\delta)\cdot s_{0,B}^2 + \delta_{pB}^2, \tag{8}$$

for some $\delta \geq 0$. On the other hand, we have: $\sigma_{\min}(\mathbf{\Sigma}_{0,B}) = \frac{1}{\|\mathbf{\Sigma}_{0,B}^{-1}\|}$ and

$$\begin{aligned}\|\mathbf{\Sigma}_{0,B}^{-1}\| &= \left\|\mathbf{\Sigma}_{1,B}^{-1} + \frac{1}{s_{1,B}^2}\mathbf{R}_B^\top\mathbf{R}_B\right\| && \text{(by definition)}\\ &\leq \|\mathbf{\Sigma}_{1,B}^{-1}\| + \frac{1}{s_{1,B}^2}\|\mathbf{R}_B^\top\mathbf{R}_B\| && \text{(by triangle inequality)}\end{aligned}$$

$$= \frac{1}{s_B^2} + \frac{Q_x^\infty}{s_{1,B}^2} \qquad \text{(by the fact } \mathbf{R}_B^\top \mathbf{R}_B = Q_x^\infty \mathbf{I}\text{)}$$

$$\leq \frac{1}{s_B^2} \cdot (1 + 1/\eta^2). \tag{9}$$

The last inequality uses the assumed lower bound (2) on $s_{1,B}/s_B$. Therefore, using (9), we see that the sampleability condition (8) is implied by the desired lower bound $\frac{s_B^2}{1+1/\eta^2} \geq (1+\delta)s_{0,B}^2 + \delta_{pB}^2$, whereas the assumed condition (3) on $s_{0,B}/s_B$ is equivalent to the lower bound $\frac{s_B^2}{1+1/\eta^2} \geq (1+\delta)s_{0,B}^2 + \epsilon_p s_B^2$. Comparing these assumed and desired lower bounds on $\frac{s_B^2}{1+1/\eta^2}$, we see that the former implies the latter if $\epsilon_p s_B^2 \geq \delta_{pB}^2$, which in turn is equivalent to the assumed lower bound (5) on $s_B$, as required.

**Analysis of $\mathcal{B}$'s advantage**

**Case 1: $b = 1$; Part 1: analysis of SimHintMLWE oracle.** Next, to analyze the advantage of $\mathcal{B}$ against MLWE-PRF, we consider first the case that $b = 1$. In this case, consider a query $\mathbf{R}_j$ by $\mathcal{A}$ to SimHintMLWE. The vector $\mathbf{c}'_j$ returned by the SampMLWE has the form

$$\mathbf{c}'_j = \mathbf{A}_j \mathbf{k} + \mathbf{e}'_j, \tag{10}$$

where $\mathbf{A}_j$ is uniformly random in $R_q^{h \times N}$ and $\mathbf{e}'_j$ is sampled from the distribution $\mathcal{D}_{\mathbb{Z}^{hd}, s_0^2 \mathbf{I}}$. Hence, we have

$$\mathbf{c}_j = \mathbf{A}_j \mathbf{k} + \mathbf{e}_j, \tag{11}$$

where $\mathbf{e}_j := \mathbf{e}'_j + \mathbf{t}_j$ is sampled from the distribution $\mathcal{D}_{\mathsf{conv}} := \mathcal{D}_{\mathbb{Z}^{Nd}, s_0^2 \mathbf{I}} + \mathcal{D}_{\mathbb{Z}^{Nd}, \mathbf{\Sigma}_{0,j} - s_0^2 \mathbf{I}, \mathbf{d}_j}$. By the convolution Lemma 2, the distribution $\mathcal{D}_{\mathsf{conv}}$ is within statistical distance $\leq 2\epsilon$ of $\mathcal{D}_{\mathbb{Z}^{Nd}, \mathbf{\Sigma}_{0,j}, \mathbf{d}_j}$, assuming that the following smoothing condition holds:

$$\sqrt{\mathbf{\Sigma}_{3,j}} \geq \eta_\epsilon(\mathbb{Z}^{Nd}), \text{ where } \mathbf{\Sigma}_{3,j} := \frac{1}{s_0^2}\mathbf{I} + \left(\mathbf{\Sigma}_{0,j} - s_0^2 \mathbf{I}\right)^{-1}. \tag{12}$$

We show in Claim 1 below that (12) is satisfied. Therefore, by the "one-matrix hint" conditional distribution Lemma 5, for each $j \in [Q_{\mathsf{M}}]$, the pair $(\mathbf{c}_j, \mathbf{h}_j)$ returned by the SimHintMLWE simulator is within statistical distance $\leq 2\epsilon$ of the distribution of the output returned by the SampHintMLWE oracle in the real Hint-MLWE-PRF$^1$ game. Note that the $j$'th response hint $\mathbf{h}_j$ provides a hint about the $j$'th secret error $\mathbf{e}_j$ only, and does not depend on any other secret errors $\mathbf{e}_i$ and the secret errors $\mathbf{e}_i$'s are independent. So we can apply the "one-matrix hint" conditional distribution Lemma 5 inductively for each query $j \in [Q_{\mathsf{M}}]$. Hence, we conclude that the joint distribution of the tuples $((\mathbf{A}_j, \mathbf{c}_j, \mathbf{h}_j)_{j \in [Q_{\mathsf{M}}]})$ in the view of $\mathcal{A}$ simulated by $\mathcal{B}$ in the MLWE-PRF$^1$ game is within statistical distance $\leq 2Q_{\mathsf{M}}\epsilon$ of the distribution of $\mathcal{A}$'s view in the real Hint-MLWE-PRF$^1$ game.

**Case 1: $b = 1$; Part 2: analysis of SimHintPRF oracle.** Similarly, in SimHintPRF, the PRF output $\mathbf{y}'_B$ returned by the SampPRF oracle for a query $x$ has the form

$$\mathbf{y}'_B = \mathbf{B}_x \mathbf{k} + \mathbf{e}'_B, \tag{13}$$

where $\mathbf{B}_x = \mathsf{H}(x)$ and $\mathbf{e}'_B$ is sampled from the distribution $\mathcal{D}_{\mathbb{Z}^{hd}, \hat{\mathbf{\Sigma}}_{0,B}}$. Hence, we have

$$\mathbf{y}_B = \mathbf{B}_x \mathbf{k} + \mathbf{e}_B, \tag{14}$$

where $\mathbf{e}_B := \mathbf{e}'_B + \mathbf{t}_B$ is sampled from the distribution $\mathcal{D}_{\mathsf{convB}} := \mathcal{D}_{\mathbb{Z}^{hd}, \hat{\mathbf{\Sigma}}_{0,B}} + \mathcal{D}_{\mathbb{Z}^{hd}, \mathbf{\Sigma}_{0,B} - \hat{\mathbf{\Sigma}}_{0,B}, \mathbf{d}_B}$. By the convolution Lemma 2, the distribution $\mathcal{D}_{\mathsf{convB}}$ is within statistical distance $\leq 2\epsilon$ of $\mathcal{D}_{\mathbb{Z}^{hd}, \mathbf{\Sigma}_{0,B}, \mathbf{d}_B}$, assuming that the following smoothing condition holds:

$$\sqrt{\mathbf{\Sigma}_{3,B}} \geq \eta_\epsilon(\mathbb{Z}^{hd}), \text{ where } \mathbf{\Sigma}_{3,B}^{-1} := \hat{\mathbf{\Sigma}}_{0,B}^{-1} + \left(\mathbf{\Sigma}_{0,B} - \hat{\mathbf{\Sigma}}_{0,B}\right)^{-1}. \tag{15}$$

We show below in Claim 1 that (15) is satisfied. Therefore, by the matrix hint conditional distribution Lemma 5 with respect to the $Q_x^\infty$-hint matrix $\mathbf{R}_B := -[\mathbf{I}, \cdots, \mathbf{I}]^\top \in \mathbb{Z}^{Q_x^\infty hd \times hd}$ (since all $Q_x^\infty$ hints $\mathbf{h}_{B,j}$ for the same $x$ are with respect to the same error $\mathbf{e}_B$), we conclude that the collection of triples $(\mathbf{B}_x, \mathbf{y}_B, \mathbf{h}_{B,j})_{j\in[Q_x]}$ returned by the $\mathsf{SimHintPRF}$ simulator for each queried $x$ is within statistical distance $\leq 2\epsilon$ of the distribution of the output returned by the $\mathsf{SampHintPRF}$ oracle in the real $\mathsf{Hint\text{-}MLWE\text{-}PRF}^1$ game. Adding up over all $\leq Q$ $x$ queries, we conclude that the joint distribution of the $(\mathbf{B}_x, \mathbf{y}_B, \mathbf{h}_{B,j})_{j\in[Q_x]}$ tuples in the view of $\mathcal{A}$ simulated by $\mathcal{B}$ in the $\mathsf{MLWE\text{-}PRF}^1$ game is within statistical distance $\leq 2Q\epsilon$ of the distribution of $\mathcal{A}$'s view in the real $\mathsf{Hint\text{-}MLWE\text{-}PRF}^1$ game.

Combining Part 1 and Part 2, the overall view of $\mathcal{A}$ simulated by $\mathcal{B}$ in the $\mathsf{MLWE\text{-}PRF}^1$ game is within statistical distance $\leq 2(Q + Q_\mathsf{M})\epsilon$ of the distribution of $\mathcal{A}$'s view in the real $\mathsf{Hint\text{-}MLWE\text{-}PRF}^1$ game.

**Case 2:** $b = 0$**.** We now consider the second case that $b = 0$. In this case, the response vectors $\mathbf{y}'_B$ and $\mathbf{c}'_j$ are sampled from the uniform distribution on $R_q^h$ and $R_q^N$, respectively. Hence, $\mathbf{y}_B = \mathbf{y}'_B + \mathbf{t}_B$ and $\mathbf{c}_j = \mathbf{c}'_j + \mathbf{t}_j$ are also uniformly distributed on $R_q^h$ and $R_q^N$, respectively. They are independent of the hint vectors $\mathbf{h}_j$ and $\mathbf{h}_{B,j}$, while the marginal distribution of the hint vectors remain the same as in the $b = 1$ game. Therefore, the distribution of the view of $\mathcal{B}$ simulated by $\mathcal{A}$ in the $\mathsf{MLWE\text{-}PRF}^0$ game is exactly equal to the distribution of $\mathcal{B}$'s view in the real $\mathsf{Hint\text{-}MLWE\text{-}PRF}^0$ game.

From the above analysis for Case 1 and Case 2, we conclude that $\mathsf{Adv}_\mathcal{B}^{\mathsf{MLWE\text{-}PRF}}(\lambda) \geq \mathsf{Adv}_\mathcal{A}^{\mathsf{Hint\text{-}MLWE\text{-}PRF}}(\lambda) - 2(Q + Q_\mathsf{M}) \cdot \epsilon$, which is the claimed advantage bound. What remains to prove is the smoothing conditions.

*Claim 1.* The smoothing conditions (12) and (15) are satisfied based on the assumptions in the Theorem statement.

*Proof of Claim 1:* By Lemma 4, the condition (12) holds if

$$\|\mathbf{\Sigma}_{3,j}^{-1}\| \leq \eta_\epsilon(\mathbb{Z}^{Nd})^{-2}. \tag{16}$$

From (6), we have $\sigma_{\min}(\mathbf{\Sigma}_{0,j} - s_0^2\mathbf{I}) \geq \delta \cdot s_0^2$. Therefore, the triangle inequality gives

$$\|\mathbf{\Sigma}_{3,j}^{-1}\| \leq \frac{1}{s_0^2} + \|(\mathbf{\Sigma}_{0,j} - s_0^2\mathbf{I})^{-1}\| = \frac{1}{s_0^2} + \frac{1}{\sigma_{\min}(\mathbf{\Sigma}_{0,j} - s_0^2\mathbf{I})} \leq \frac{1 + 1/\delta}{s_0^2}. \tag{17}$$

Combining (17) and (16), we conclude that (12) is satisfied if $s_0 \geq \sqrt{1 + 1/\delta} \cdot \eta_\epsilon(\mathbb{Z}^{Nd})$, and the latter condition is implied by the assumed lower bound (4) on $s_0$, using Lemma 3, as required.

Similarly, by Lemma 4, the condition (15) holds if

$$\|\mathbf{\Sigma}_{3,B}^{-1}\| \leq \eta_\epsilon(\mathbb{Z}^{hd})^{-2}. \tag{18}$$

From (8), we have $\sigma_{\min}(\mathbf{\Sigma}_{0,B} - \hat{\mathbf{\Sigma}}_{0,B}) \geq \delta \cdot s_{0,B}^2$. Therefore, the triangle inequality gives

$$\|\mathbf{\Sigma}_{3,B}^{-1}\| \leq \|\hat{\mathbf{\Sigma}}_{0,B}^{-1}\| + \|(\mathbf{\Sigma}_{0,B} - \hat{\mathbf{\Sigma}}_{0,B})^{-1}\| \tag{19}$$

$$= \frac{1}{\sigma_{\min}(\hat{\mathbf{\Sigma}}_{0,B})} + \frac{1}{\sigma_{\min}(\mathbf{\Sigma}_{0,B} - \hat{\mathbf{\Sigma}}_{0,B})} \tag{20}$$

$$\leq \frac{\hat{\rho}_{0,B} + 1/\delta}{s_{0,B}^2}. \tag{21}$$

Combining (21) and (18), we conclude that (15) is satisfied if $s_{0,B} \geq \sqrt{\hat{\rho}_{0,B} + 1/\delta} \cdot \eta_\epsilon(\mathbb{Z}^{hd})$, and the latter condition is implied by the assumed lower bound (4) on $s_{0,B}$, using Lemma 3, as required. □

### 3.3 Hardness Reduction from MLWE to MLWE-PRF

The final piece of our analyses on the security of $\mathsf{MLWE\text{-}PRF\text{-}RU}$ is to reduce from $\mathsf{MLWE}$ to $\mathsf{MLWE\text{-}PRF}$. This reduction is contingent upon the concrete choice of the mapping $\mathsf{H}$ and we discuss two natural options in this section.

**$\mathsf{H}$ is a random oracle.** It is easy to see from the description of MLWE-PRF in Figure 4 that if $\mathsf{H}$ is a random oracle, then SampPRF simply outputs MLWE samples with error distribution $\chi_{0,B}$. Assuming $\chi_0 := \mathcal{D}_{\mathbb{Z}^d,s_0}$ and $\chi_{0,B} := \mathcal{D}_{\mathbb{Z}^d,s_{0,B}}$ with $s_0 \le s_{0,B}$ (without loss of generality), we can construct samples of SampPRF using regular MLWE samples with error distribution $\chi_0 := \mathcal{D}_{\mathbb{Z}^d,s_0}$ by adding an error term with Gaussian width $\sqrt{s_{0,B}^2 - s_0^2}$ as similarly done before. If $s_0 \ge s_{0,B}$, then we can similarly construct MLWE samples with error distribution $\chi_0$ using MLWE samples with error distribution $\chi_{0,B}$. We therefore get the following theorem.

**Theorem 3 (MLWE $\to$ MLWE-PRF for random oracle $\mathsf{H}$).** *Let $\mathsf{H}$ be a random oracle, and $\chi_0 := \mathcal{D}_{\mathbb{Z}^d,s_0}$ and and $\chi_{0,B} := \mathcal{D}_{\mathbb{Z}^d,s_{0,B}}$. Let $\hat{s} := \min\{s_0, s_{0,B}\}$. The $\mathsf{MLWE\text{-}PRF}_{\mathsf{H},Q_\mathsf{P},Q_\mathsf{M},q,m,h,L,\chi_0,\chi_{0,B},\chi_k}$ assumption holds if $\mathsf{MLWE}_{Q_\mathsf{P}+Q_\mathsf{M},q,m,1,\hat{\chi},\chi_k}$ assumption holds for $\hat{\chi} := \mathcal{D}_{\mathbb{Z}^d,\hat{s}}$. More precisely, for any PPT adversary $\mathcal{A}$ against the $\mathsf{MLWE\text{-}PRF}_{\mathsf{H},Q_\mathsf{P},Q_\mathsf{M},q,m,h,L,\chi_0,\chi_{0,B},\chi_k}$ problem, there exists a PPT adversary $\mathcal{B}$ against the $\mathsf{MLWE}_{Q_\mathsf{H}+Q_\mathsf{P}+Q_\mathsf{M},q,m,1,\hat{\chi},\chi_k}$ problem with $Q_\mathsf{H}$ being the number of random oracle $\mathsf{H}$ queries, such that*

$$\mathsf{Adv}^{\mathsf{MLWE\text{-}PRF}}_{\mathcal{A}}(\lambda) \le \mathsf{Adv}^{\mathsf{MLWE}}_{\mathcal{B}}(\lambda).$$

**$\mathsf{H}$ is defined based on BP14 PRF.** As discussed in [BP14], the BMLR13 PRF [BLMR13] can be seen as a special case of BP14 PRF. Therefore, we focus here on BP14 PRF instantiation; the results extend to BMLR13 PRF as a special case.

**Definition 13 (Notations for BP14 PRF).** *Let $\mathfrak{l} := \lceil \log_2 q \rceil$. The public parameters are two random matrices $\mathbf{A}_0, \mathbf{A}_1 \xleftarrow{\$} R_q^{m \times m\mathfrak{l}}$. The BP14 PRF computation involves a gadget matrix $\mathbf{G}$ and its (non-linear) inverse computation $\mathbf{G}^{-1}\colon R_q^{m\times m\mathfrak{l}} \to R_q^{m\mathfrak{l}\times m\mathfrak{l}}$ (i.e., digit decomposition).*

*Let $x = (x_0, x_1, \ldots, x_{L-1}) \in \{0,1\}^L$. For $i = 1, \ldots, L-1$, define $\mathbf{B}_{x/i} := \mathbf{G}^{-1}(\mathbf{A}_{x_i} \cdot \mathbf{G}^{-1}(\mathbf{A}_{x_{i+1}} \cdots \mathbf{G}^{-1}(\mathbf{A}_{x_{L-1}})))$ and $x \setminus i := x_0 \cdots x_{i-1}$ the $i$-bit prefix of $x$. Therefore, $\mathbf{B}_{x/i} = \mathbf{G}^{-1}(\mathbf{A}_{x_i} \cdot \mathbf{B}_{x/i+1})$ and $\mathbf{G} \cdot \mathbf{B}_{x/i} = \mathbf{A}_{x_i} \cdot \mathbf{B}_{x/i+1}$.*

*For given $\mathbf{A}_0, \mathbf{A}_1$ and $x$, let $\chi_{\mathbf{A_0},\mathbf{A_1},x,s_0}$ be the distribution that samples $\mathbf{e}_{x,0}, \ldots, \mathbf{e}_{x,L-1} \xleftarrow{\$} \chi_0^{m\mathfrak{l}}$ and outputs $\mathbf{e}_x^\top := \sum_{j=0}^{L-2} \mathbf{e}_{x,j}^\top \mathbf{B}_{x/(j+1)} + \mathbf{e}_{x,L-1}^\top$. We write $\mathbf{e}_{x/i}^\top := \sum_{j=i}^{L-2} \mathbf{e}_{x,j}^\top \mathbf{B}_{x/(j+1)} + \mathbf{e}_{x,L-1}^\top$. Note that the same $\mathbf{e}_{x,j-1} = \mathbf{e}_{x',j-1}$ is used in sampling $\mathbf{e}_x, \mathbf{e}_{x'}$ for $x, x'$ with the same $j$-bit prefix, i.e. if $x \setminus j = x' \setminus j$.*

*The message mapping $\mathsf{H}^{\mathsf{bp}}$ for BP14 PRF is defined as follows.*

$$\mathsf{H}^{\mathsf{bp}}_{\mathbf{A}_0,\mathbf{A}_1}\colon x \mapsto \mathbf{B}^{\mathsf{bp}}_x, \text{ s.t. } (\mathbf{B}^{\mathsf{bp}}_x)^\top := \mathbf{A}_{x_0} \cdot \mathbf{G}^{-1}(\mathbf{A}_{x_1} \cdots \mathbf{G}^{-1}(\mathbf{A}_{x_{L-2}} \cdot \mathbf{G}^{-1}(\mathbf{A}_{x_{L-1}}))). \tag{22}$$

Observe that $(\mathsf{H}^{\mathsf{bp}}_{\mathbf{A}_0,\mathbf{A}_1}(x) \cdot \mathbf{k})^\top = \mathbf{k}^\top \cdot \mathbf{A}_{x_0} \mathbf{B}_{x/1}$.

We now recall the pseudorandomness result from [BP14] most relevant to us.

**Lemma 6 (MLWE $\to$ BP14-PRF [BP14, Theorem 3.8]).** *The function family $F_{\mathbf{A}_0,\mathbf{A}_1,\mathbf{k}}(x) := \mathsf{H}^{\mathsf{bp}}_{\mathbf{A}_0,\mathbf{A}_1}(x) \cdot \mathbf{k} + \mathbf{e}_x$ is pseudorandom for $\mathbf{A}_0, \mathbf{A}_1 \xleftarrow{\$} R_q^{m\times m\mathfrak{l}}$, $\mathbf{k} \xleftarrow{\$} R_q^m$ and $\mathbf{e}_x \xleftarrow{\$} \chi_{\mathbf{A_0},\mathbf{A_1},x,s_0}$ if multi-secret MLWE problem (i.e., MLWE with secret matrix $\mathbf{S}$) is hard where each secret key vector is sampled uniformly random in $R_q^m$ and each error vector is sampled independently from $\chi_0^{m\mathfrak{l}}$.*

The high-level idea for the proof of Lemma 6 in [BP14] is to "peel off" each layer of computation in $\mathsf{H}(x)$ one by one using the observation that $\mathbf{k}^\top \cdot \mathbf{A}_{x_i} \mathbf{B}_{x/(i+1)} + \mathbf{e}_{x/i}^\top = (\mathbf{k}^\top \cdot \mathbf{A}_{x_i} + \mathbf{e}_{x,i}^\top) \cdot \mathbf{B}_{x/(i+1)} + \mathbf{e}_{x/(i+1)}^\top$. Assuming MLWE hardness, we can replace the term inside the parentheses by a uniformly random vector, say $\mathbf{u}_{x\setminus(i+1)}^\top$, over $R_q$, that is independently sampled for each $(i+1)$-bit prefix $x \setminus (i+1)$. We can then decompose $\mathbf{u}_{x\setminus(i+1)}^\top = \mathbf{k}_{x\setminus(i+1)}^\top \cdot \mathbf{G} + \mathbf{v}_{i+1}^\top$ for some $\mathbf{v}_{i+1}^\top$ uniform in the quotient group $R_q^{m\mathfrak{l}}/(\mathbf{G}R_q^m)$ and *independent* of $\mathbf{k}_{x\setminus(i+1)}^\top$ (these $\mathbf{k}_{x\setminus(i+1)}^\top$, sampled independently for each $(i+1)$-bit prefix $x \setminus (i+1)$, make up the (multi-secret) MLWE secret matrix $\mathbf{S}$). Plugging in $\mathbf{u}_{x\setminus(i+1)}^\top = \mathbf{k}_{x\setminus(i+1)}^\top \cdot \mathbf{G} + \mathbf{v}_{i+1}^\top$ back to the above expression, we will get a similarly formed expression for the next index $i+1$. We can therefore continue to inductively peel off all layers until at the end we are left with a uniformly random vector $\mathbf{u}_{x\setminus L}^\top = \mathbf{u}_x^\top$ which is independently sampled for each $x \in \{0,1\}^L$, i.e. a uniformly random function.

From this blueprint, the MLWE $\to$ BP14-PRF reduction naturally extends to a MLWE $\to$ MLWE-PRF reduction as follows. In the first hybrid, while peeling off the first layer ($i = 0$) of the PRF samples as

discussed above, with $\mathbf{k}^\top \cdot \mathbf{A}_{x_0} + \mathbf{e}_{x,0}^\top$ simulated from the MLWE oracle samples, we additionally simulate the SampMLWE outputs in MLWE-PRF by requesting more samples from the MLWE oracle with respect to the same secret $\mathbf{k}$. So at this first layer, hardness of MLWE implies we can replace both PRF samples and SampMLWE outputs by independent uniformly random outputs. Once the additional SampMLWE outputs are replaced by uniform, they remain uniform for the remaining hybrids, and the remaining hybrids can proceed similarly as in [BP14]. This high-level idea leads to the following theorem, whose formal proof is provided in Appendix D.1.

**Theorem 4** (MLWE $\to$ MLWE-PRF **for BP14** H)**.**
*Let* $\mathsf{H}^{\mathsf{bp}}$ *be the BP14 function family with L-bit input and error distribution* $\chi_{\mathbf{A_0},\mathbf{A_1},x,s_0}$ *as in Def. 13. Let* $\chi_k := \mathcal{U}(R_q)$, $\chi_0 := \mathcal{D}_{\mathbb{Z}^d,s_0}$ *and* $\chi_{0,B} := \chi_{\mathbf{A_0},\mathbf{A_1},x,s_0}$. *The* $\mathsf{MLWE\text{-}PRF}_{\mathsf{H}^{\mathsf{bp}},Q_\mathsf{P},Q_\mathsf{M},q,m,h,L,\chi_0,\chi_{0,B},\chi_k}$ *assumption holds if the* $\mathsf{MLWE}_{2m\mathfrak{l}+Q_\mathsf{M},q,m,1,\chi_0,\chi_k}$ *and* $\mathsf{MLWE}_{2m\mathfrak{l},q,m,Q_\mathsf{P},\chi_0,\chi_k}$ *assumptions hold.*

*More precisely, for any* PPT *adversary* $\mathcal{A}$ *against the* $\mathsf{MLWE\text{-}PRF}_{\mathsf{H}^{\mathsf{bp}},Q_\mathsf{P},Q_\mathsf{M},q,m,h,L,\chi_0,\chi_{0,B},\chi_k}$ *problem, there exist* PPT *adversaries* $\mathcal{B},\overline{\mathcal{B}}$ *against the* $\mathsf{MLWE}_{2m\mathfrak{l}+Q_\mathsf{M},q,m,1,\chi_0,\chi_k}$ *and* $\mathsf{MLWE}_{2m\mathfrak{l},q,m,Q_\mathsf{P},\chi_0,\chi_k}$ *problems respectively, such that*

$$\mathsf{Adv}^{\mathsf{MLWE\text{-}PRF}}_{\mathcal{A}}(\lambda) \le \mathsf{Adv}^{\mathsf{MLWE}}_{\mathcal{B}}(\lambda) + (L-1)\cdot \mathsf{Adv}^{\mathsf{MLWE}}_{\overline{\mathcal{B}}}(\lambda)\,.$$

Note that for each fixed $x$, the PRF error distribution $\chi_{0,B} := \chi_{\mathbf{A_0},\mathbf{A_1},x,s_0}$ in Theorem 4 *marginalized* for a single PRF input $x$ is a convolution of Gaussian distributions scaled by short matrices, and hence is statistically close to a non-spherical Gaussian distribution, by standard discrete Gaussian convolution results. Our MLWE-PRF $\to$ Hint-MLWE-PRF reduction Theorem 2 supports such non-spherical Gaussian distributions for $\chi_{0,B}$. However, Theorem 2 is currently not fully compatible with Theorem 4. This is because in the *joint* distribution in Theorem 4 of two PRF errors $(\mathbf{e}_x^\top, \mathbf{e}_{x'}^\top)$ for distinct inputs $x, x'$ with a common prefix, there is a correlation between the errors (due to the error vectors sharing a common additive error term corresponding to the common prefix), while such two errors are assumed independently sampled in Theorem 2. Extending Theorem 4 to support independent error samples or Theorem 2 to support correlated error samples for distinct $x$ is currently an open question for $\mathsf{H}^{\mathsf{bp}}$ in the standard model. Still, we view Theorem 4 as a bonus qualitative evidence for the hardness of MLWE-PRF with $\mathsf{H}^{\mathsf{bp}}$ in the standard model, in addition to our hardness result Theorem 3 for MLWE-PRF with a random oracle H and independent error samples. We also remark that both MLWE $\to$ MLWE-PRF reductions in Theorems 3 and 4 are tight.

## 4 LeOPaRd : Our Lattice-Based VPOPRF Proposal

At a very high-level, our goal is to realize the 2HashDH OPRF idea [JKK14] in the lattice setting as first done by Albrecht et al. [ADDS21]. Our construction ensures that a client with an input/tag pair $(x,t) \in \{0,1\}^L$ can recover the PRF value $\lfloor \mathbf{B}_{x,t}\mathbf{k} \rceil_p$ after interacting with the server, where $\mathbf{B}_{x,t} := \mathsf{H}(x,t)$, for some *message mapping* H, and $\mathbf{k}$ is the secret key of the server. We discuss potential instantiations of the function H and its relation to the existing lattice-based PRF schemes in Appendix C.2. As discussed before, our OPRF construction, LeOPaRd, is round-optimal (see Remark 3), and supports partial obliviousness and verifiability.

We would like the OPRF evaluation to begin with the client linearly encrypting the matrix $\mathbf{B}_{x,t}$ such that $\mathbf{C}_x := \mathbf{R}\mathbf{A}_r + \mathbf{B}_{x,t}$, followed by the server computing $\mathbf{u}_x := \mathbf{C}_x\mathbf{k} + \mathbf{e}'_s$ for some error vector $\mathbf{e}'_s$ (so that $\mathbf{u}_x$ contains the term $\mathbf{B}_{x,t}\mathbf{k}$). However, to minimize the client's control over $\mathbf{C}_x$ so that $\mathbf{u}_x$ does not leak server's secret key, we want the client to commit to the pair $\mathbf{R}$, and generate the matrix $\mathbf{A}_r$ via a random oracle using the resulting commitment $\mathbf{c}_r$.

This way, the client is forced to pick their randomness $\mathbf{R}$ *before* seeing the random matrix $\mathbf{A}_r$. We show that the adversary can later decide adaptively on the input $x$ without harming security. In fact, this not only suffices for reducing the pseudorandomness of LeOPaRd to our new MLWE-PRF-RU assumption (Definition 10), but it also allows us to utilize preprocessing (Figure 8) for improving the efficiency during the online phase of our protocol (Figure 9).

During the preprocessing phase, the client can sample multiple randomness $\mathbf{R}_i$ and commit to them as $\mathbf{c}_{r,i}$, which the server can use to derive the corresponding $\mathbf{A}_{r,i}$ and $\mathbf{v}_{k,i}$ values before the online phase

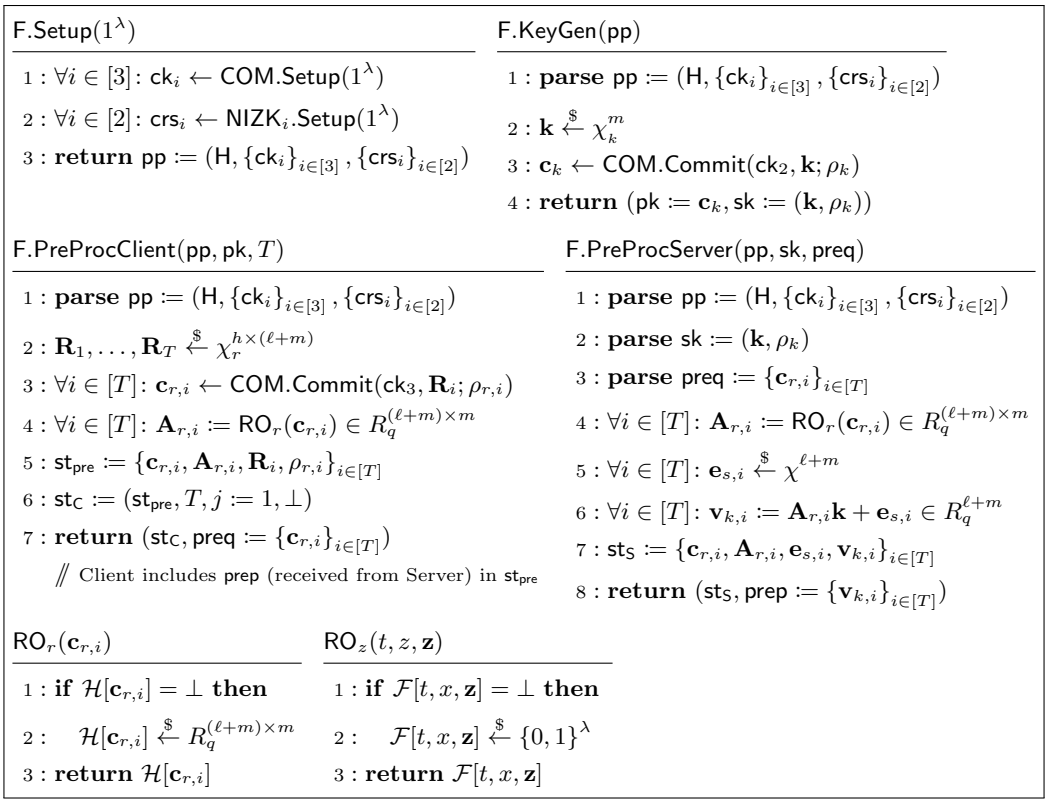


Fig. 8: Setup, preprocessing and random oracles of LeOPaRd.

starts (we note that these $\mathbf{v}_{k,i}$ values are given to the client as well). Then, during the online phase, the client consumes the $\mathbf{R}_j$ values one by one (where $j$ denotes the counter that the client keeps in Figure 9). Concretely, the client uses $\mathbf{R}_j$ and $\mathbf{A}_{r,j}$ to derive the masked input $\mathbf{C}_x := \mathbf{R}_j\mathbf{A}_{r,j} + \mathbf{B}_{x,t}$, and additionally, computes a commitment to the input $x$, that we denote as $\mathbf{d}_x$[11], and shares both $\mathbf{C}_x$ and $\mathbf{d}_x$ with the server.

Upon receiving $\mathbf{C}_x$ and $\mathbf{d}_x$ from the client, the server responds to the client with $\mathbf{u}_x := \mathbf{C}_x\mathbf{k} + \mathbf{e}'_s$, where $\mathbf{e}'_s$ denote some appropriately distributed errors. At this point, the client can use its secret randomness $\mathbf{R}_j$ to recover the PRF output $\lfloor \mathbf{u}_x - \mathbf{R}_j\mathbf{v}_{k,j} \rceil_p = \lfloor \mathbf{B}_{x,t}\mathbf{k} + \mathbf{e}_f \rceil_p = \lfloor \mathbf{B}_{x,t}\mathbf{k} \rceil_p$, where $\mathbf{e}_f := \mathbf{e}'_s - \mathbf{R}_j\mathbf{e}_s$ is some final error term with small coefficients (relative to $q$).

To additionally achieve verifiability and protect against malicious adversaries, we require the client and server to prove the well-formedness of their computations with NIZK proofs.

Our full POPRF construction is given in Figures 8 and 9, and the NIZK relations of client and server proofs, i.e., $\pi_c$ and $\pi_s$, are given in Figure 10. In the protocol description, $\mathsf{NIZK}_1$ and $\mathsf{NIZK}_2$ denote the NIZK argument system of the client and server, respectively. We also summarize the main notations/parameters in Table 2. We instantiate the commitment scheme COM using Regev-style encryption [Reg05], which can also be seen as an extractable version of the BDLOP commitment [BDL+18]. Given that such an encryption is quite standard by now, we defer the details to Appendix F.

*Remark 2.* In our LeOPaRd description in Figure 9, the client's first move F.Request has the server's public key pk as an input as we follow the formal model from [TCR+22]. However, the client does not actually make use of pk in F.Request, in fact, only ever uses it in verifying $\pi_s$. There are important applications of

[11] We note that we require the commitment $\mathbf{d}_x$ just to make the security proof of Theorem 5 go through, since during the proof we need to extract the input $x$.

$\mathsf{F.Request}(\mathsf{pp}, \mathsf{pk}, t, x, \mathsf{st_C})$

1 : **parse** $\mathsf{pp} \coloneqq (\mathsf{H}, \{\mathsf{ck}_i\}_{i\in[3]}, \{\mathsf{crs}_i\}_{i\in[2]})$
2 : **parse** $\mathsf{st_C} \coloneqq (\mathsf{st_{pre}}, T, j, \cdot)$
3 : $\mathsf{st_{pre}} \coloneqq \{\mathbf{c}_{r,i}, \mathbf{A}_{r,i}, \mathbf{R}_i, \mathbf{v}_{k,i}, \rho_{r,i}\}_{i\in[T]}$
4 : **if** $j > T$ **then return** $\perp$
5 : $\mathbf{B}_{x,t} \coloneqq \mathsf{H}(x,t) \in R_q^{h\times m}$
6 : $\mathbf{C}_x \coloneqq \mathbf{R}_j\mathbf{A}_{r,j} + \mathbf{B}_{x,t} \in R_q^{h\times m}$
7 : $\mathbf{d}_x \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_1, x; \rho_x)$
8 : $\mathsf{stmt}_1 \coloneqq (\mathbf{c}_{r,j}, \mathbf{C}_x, \mathbf{d}_x, \mathsf{ck}_1, \mathsf{ck}_3, \mathbf{A}_{r,j}, \mathsf{H}, t, j)$
9 : $\mathsf{wit}_1 \coloneqq (\mathbf{R}_j, x, \rho_{r,j}, \rho_x)$
10 : $\pi_c \leftarrow \mathsf{NIZK}_1.\mathsf{P}(\mathsf{crs}_1, \mathsf{stmt}_1, \mathsf{wit}_1)$
11 : $\mathsf{st}_j \coloneqq (t, x, \mathbf{C}_x, \mathsf{pk})$
12 : $\mathsf{st_C} \coloneqq (\mathsf{st_{pre}}, T, j+1, \mathsf{st}_j)$
13 : $\mathsf{req} \coloneqq (\mathbf{d}_x, \mathbf{C}_x, \pi_c, j)$
14 : **return** $(\mathsf{st_C}, \mathsf{req})$

$\mathsf{F.Finalize}(\mathsf{pp}, \mathsf{rep}, \mathsf{st_C})$

1 : **parse** $\mathsf{pp} \coloneqq (\mathsf{H}, \{\mathsf{ck}_i\}_{i\in[3]}, \{\mathsf{crs}_i\}_{i\in[2]})$
2 : **parse** $\mathsf{st_C} \coloneqq (\mathsf{st_{pre}}, T, j+1, \mathsf{st}_j)$
3 : **parse** $\mathsf{st_{pre}} \coloneqq \{\mathbf{c}_{r,i}, \mathbf{A}_{r,i}, \mathbf{R}_i, \mathbf{v}_{k,i}, \rho_{r,i}\}_{i\in[T]}$
4 : **parse** $\mathsf{st}_j \coloneqq (t, x, \mathbf{C}_x, \mathbf{c}_k)$
5 : **parse** $\mathsf{rep} \coloneqq (\mathbf{u}_x, \pi_s)$
6 : $\mathsf{stmt}_2 \coloneqq (\mathbf{u}_x, \mathbf{C}_x, \mathbf{c}_k, \mathsf{ck}_2, \mathbf{v}_{k,j}, \mathbf{A}_{r,j})$
7 : **if** $\mathsf{NIZK}_2.\mathsf{V}(\mathsf{crs}_2, \pi_s, \mathsf{stmt}_2) \neq 1$ **then**
8 : **return** $\perp$
9 : $\mathbf{z} \coloneqq \lfloor \mathbf{u}_x - \mathbf{R}_j\mathbf{v}_{k,j} \rceil_p \in R_p^h \quad /\!/ = \lfloor \mathbf{B}_{x,t}\mathbf{k} \rceil_p$
10 : $y \coloneqq \mathsf{RO}_z(t, x, \mathbf{z})$
11 : **return** $y$

$\mathsf{F.BlindEval}(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}, t, \mathsf{req}, \mathsf{st_S})$

1 : **parse** $\mathsf{pp} \coloneqq (\mathsf{H}, \{\mathsf{ck}_i\}_{i\in[3]}, \{\mathsf{crs}_i\}_{i\in[2]})$
2 : **parse** $\mathsf{pk} \coloneqq \mathbf{c}_k, \mathsf{sk} \coloneqq (\mathbf{k}, \rho_k)$
3 : **parse** $\mathsf{st_S} \coloneqq \{\mathbf{c}_{r,i}, \mathbf{A}_{r,i}, \mathbf{e}_{s,i}, \mathbf{v}_{k,i}\}_{i\in[T]}$
4 : **parse** $\mathsf{req} \coloneqq (\mathbf{d}_x, \mathbf{C}_x, \pi_c, j)$
5 : $\mathsf{stmt}_1 \coloneqq (\mathbf{c}_{r,j}, \mathbf{C}_x, \mathbf{d}_x, \mathsf{ck}_1, \mathsf{ck}_3, \mathbf{A}_{r,j}, \mathsf{H}, t, j)$
6 : **if** $\mathsf{NIZK}_1.\mathsf{V}(\mathsf{crs}_1, \pi_c, \mathsf{stmt}_1) \neq 1$ **then**
7 : **return** $\perp$
8 : $\mathbf{e}'_s \xleftarrow{\$} \chi_1^h$
9 : $\mathbf{u}_x \coloneqq \mathbf{C}_x\mathbf{k} + \mathbf{e}'_s \in R_q^h$
10 : $\mathsf{stmt}_2 \coloneqq (\mathbf{u}_x, \mathbf{C}_x, \mathbf{c}_k, \mathsf{ck}_2, \mathbf{v}_{k,j}, \mathbf{A}_{r,j})$
11 : $\mathsf{wit}_2 \coloneqq (\mathbf{k}, \mathbf{e}_{s,j}, \mathbf{e}'_s, \rho_k)$
12 : $\pi_s \leftarrow \mathsf{NIZK}_2.\mathsf{P}(\mathsf{crs}_2, \mathsf{stmt}_2, \mathsf{wit}_2)$
13 : **return** $\mathsf{rep} \coloneqq (\mathbf{u}_x, \pi_s)$

$\mathsf{F.Eval}(\mathsf{sk}, t, x)$

1 : **parse** $\mathsf{sk} \coloneqq (\mathbf{k}, \rho_k)$
2 : $\mathbf{B}_{x,t} \coloneqq \mathsf{H}(x,t) \in R_q^{h\times m}$
3 : $\mathbf{z} \coloneqq \lfloor \mathbf{B}_{x,t}\mathbf{k} \rceil_p \in R_p^h$
4 : $y \coloneqq \mathsf{RO}_z(t, x, \mathbf{z})$
5 : **return** $y$

Fig. 9: LeOPaRd : Our verifiable POPRF construction.

OPRFs where the client is required to only remember a password (and some common reference string, which may be embedded in application software) such as password-authenticated key exchange (PAKE) [JKX18] and password-protected secret sharing (PPSS) [JKK14, JKKX16]. LeOPaRd can support such applications in the password-only model using ideas as in [JKK14], where the authenticity of the server's public key is *not* assumed during the online OPRF evaluation (i.e., no PKI is required).

*Remark 3 (Preprocessing).* We note that unlike prior works [APRR24, HKL+25, BDFH25, DDT25] that require preprocessing in order to reduce their online rounds, our protocol given in Figure 9 is round-optimal by default (i.e., we can simply bring computation of $\mathbf{c}_r$ and $\mathbf{v}_k$ from preprocessing to inside F.Request and F.BlindEval, respectively). Hence, the sole reason for us to employ preprocessing is to achieve smaller online communication size (not smaller number of communication rounds).

*Remark 4 (Threshold VPOPRF).* Although we presented (for the sake of simplicity) our construction from Figures 8 and 9 in a single server setting, we note that it easily extends to $n$-out-of-$n$ setting, as

The server proof $\pi_s$ proves the following,

$$\pi_s \coloneqq \left\{ \begin{array}{l} \mathfrak{Z}(\mathbf{u}_x, \mathbf{C}_x, \mathbf{c}_k, \mathsf{ck}_2, \mathbf{v}_k, \mathbf{A}_r, \beta_k, \beta, \beta_1);\ \mathfrak{K}(\mathbf{k}, \mathbf{e}_s, \mathbf{e}'_s, \rho_k) \mid \|\mathbf{k}\|_\infty \leq \beta_k \wedge \\ \|\mathbf{e}_s\|_\infty \leq \beta \ \wedge\ \|\mathbf{e}'_s\|_\infty \leq \beta_1 \ \wedge\ \mathbf{c}_k = \mathsf{COM.Commit}(\mathsf{ck}_2, \mathbf{k}; \rho_k) \ \wedge \\ \mathbf{v}_k = \mathbf{A}_r\mathbf{k} + \mathbf{e}_s \bmod q \wedge \mathbf{u}_x = \mathbf{C}_x\mathbf{k} + \mathbf{e}'_s \bmod q \end{array} \right\}.$$

The client proof $\pi_c$ proves the following,

$$\pi_c \coloneqq \left\{ \begin{array}{c} \mathfrak{Z}(\mathbf{c}_r, \mathbf{C}_x, \mathbf{d}_x, \mathsf{ck}_1, \mathsf{ck}_3, \mathbf{A}_r, \mathsf{H}, t, j, \beta_r);\ \mathfrak{K}(\mathbf{R}, x, \rho_r, \rho_x) \mid \|\mathbf{R}\|_\infty \leq \beta_r \wedge \\ \mathbf{B}_{x,t} \coloneqq \mathsf{H}(x,t) \ \wedge\ \mathbf{c}_r = \mathsf{COM.Commit}(\mathsf{ck}_3, \mathbf{R}; \rho_r) \wedge \\ \mathbf{d}_x = \mathsf{COM.Commit}(\mathsf{ck}_1, x; \rho_x) \wedge\ \mathbf{C}_x = \mathbf{R}\mathbf{A}_r + \mathbf{B}_{x,t} \bmod q \end{array} \right\}.$$

Fig. 10: Relations for NIZK proofs performed in LeOPaRd.

| Notation | Description |
|---|---|
| $\lambda$ | security parameter |
| $\kappa$ | correctness parameter |
| $q$ | system modulus |
| $p$ | rounding modulus |
| $d$ | ring dimension of $R = \mathbb{Z}[X]/(X^d + 1)$ |
| $\chi, \chi_1$ | server's error distributions |
| $\beta, \beta_1$ | $\ell_\infty$-norm bounds on server's errors $\mathbf{e}_s, \mathbf{e}'_s$ |
| $\chi_k$ | server's key distribution |
| $\chi_r$ | client's randomness distribution |
| $\beta_r$ | $\ell_\infty$-norm bound on client's randomness $\mathbf{R}$ |
| $s_0$ | error width in server's MLWE security |
| $m$ | dimension of server key $\mathbf{k}$ (over $R_q$) |
| $h$ | # of rows of $\mathbf{B}_x$ |
| $\ell$ | client MLWE dimension parameter |
| $n_c, n_s$ | dimensions of trapdoor keys used by client, server |

Table 2: Summary of main notations/parameters used for LeOPaRd.

in [AG24], due to the key-homomorphic properties of our VPOPRF. Concretely, we can consider $n$ servers with secret and public key pairs $(\mathbf{k}_i, \mathbf{c}_{k,i})$ generated identically as in lines 2-3 of F.KeyGen in Figure 8. Analogously, during the server preprocessing, each server computes shares of $\mathbf{v}_{k,i}$ (line 6 of F.PreProcServer in Figure 8) using their secret key shares $\mathbf{k}_i$, and in blind signing, each server computes the pair $(\mathbf{u}_{x,i}, \pi_{s,i})$ identically as in lines 9-12 of F.BlindEval in Figure 9. Finally, the user replaces line 9 of F.Finalize from Figure 9 with the computation $\mathbf{z} \coloneqq \lfloor \sum_{i=1}^n \mathbf{u}_{x,i} - \mathbf{R} \sum_{i=1}^n \mathbf{v}_{k,i} \rceil_p$ in order to obtain the final OPRF value.

*Remark 5.* We note that certain NIZK proof systems such as LNP22 [LNP22] already constructs a commitment to its witness inside the NIZK proof. In fact, as discussed in Appendix F, the trapdoor commitment scheme we use is the trapdoor version of the BDLOP commitment [BDL+18] used in [LNP22]. Therefore, by unboxing the NIZK proofs, it may be possible (in certain cases) to optimize our LeOPaRd design. However, we forego such optimizations in this work as they would make the overall protocol significantly more complex.

*Remark 6.* In our MLWE-PRF-RU security reduction from Section 3, it is important to know the maximum number of OPRF evaluation queries the adversary makes with the *same* $\mathbf{B}_{x,t} = \mathsf{H}(x,t)$. In the partially oblivious setting, since the server already knows $t$, we can let the server keep track of queried tags; and either never allow the same tag be queried twice or restrict the number of queries under the same tag, e.g., to $2^{16}$. This way, the adversary would be restricted to seeing a limited number of OPRF evaluation results for a particular $\mathbf{B}_{x,t}$. In this case, the impact of the term $Q_x^\infty$ in Theorem 2 would effectively diminish.

Some applications such as Privacy Pass [DGS+18] can benefit from batching multiple queries in one go. It is not difficult to see that LeOPaRd can support this via the client/server consuming more pre-processing outputs at once. In the online phase, both the client and the server can compute a single NIZK proof attesting

to the validity of the whole batched query. When using a succinct argument system like LaBRADOR [BS23], the proof size will increase by a very small factor. For example, as discussed in [ADDG24, App. A.2], their LaBRADOR proof size increases from 45KB (for one single query) to just 79KB for a batch of 64 queries ($< 1.8\times$ increase). Therefore, the per-query communication cost of the NIZK proof drops down to a very small value. In Appendix C, we discuss options for concretely instantiating $\mathsf{H}$ (including BP14 PRF [BP14], BLMR13 PRF [BLMR13] and random oracle) and the NIZK proofs.

### 4.1 Correctness and Security Analyses

**Correctness analysis.** We upper bound the correctness error probability of our protocol in Lemma 7, in terms of the protocol parameters. Due to space limitations, we only sketch the proofs and refer the reader to Appendix D.2 for full proofs and details.

**Lemma 7.** *Fix $\kappa, h, d$. Let $B_f(\kappa, d)$ denote an upper bound on a fixed coordinate of $\mathbf{e}_f = \mathbf{e}'_s - \mathbf{R}\mathbf{e}_s$ (as in (49)) that holds except with probability $p_e \leq 2^{-(\kappa+2+\log(hd))}$. Also, assume that the function family $\mathsf{H}$ satisfies $\epsilon_u$-*uniformity *(as per Definition 19) with*

$$\epsilon_u \leq 2^{-(\kappa+2+\log(hd))}. \tag{23}$$

*Then, for any fixed $x \in \{0,1\}^L$, $2^{-\kappa}$-correctness holds (as per Definition 2) if*

$$q/p \geq 2^{\kappa+2} \cdot hd \cdot (2B_f(\kappa, d) + 1). \tag{24}$$

**Remarks:**

- **(worst-case bound)** Note that $B_f(\kappa, d)$ is also upper bounded by the worst-case bound ($p_e = 0$):
$$B_f(\kappa, d) \leq (m + \ell)d\beta\beta_r + \beta_1 \tag{25}$$
- **(bound for Gaussian distributions)** Assume that $\chi_r = \mathcal{U}(\mathbb{S}_{\beta_r})$, $\chi = \mathcal{D}_s$, and $\chi_1 = \mathcal{D}_{s_1}$. Then, using the Gaussian convolution lemma (Lemma 2) for the term $\mathbf{R}\mathbf{e}_s$ and Gaussian tail bound, we get
$$B_f(\kappa, d) \leq \sqrt{\ln(2)(\kappa + 2 + \log(hd))/\pi} \cdot \left(s_1 + \beta_r s\sqrt{(\ell + m)d}\right). \tag{26}$$
- **(bound for uniform distributions)** In practice, one may want to avoid using Gaussian distribution in the implementation. We provide the bounds for using only uniform distribution as well. Assume that $\chi_r = \mathcal{U}(\mathbb{S}_{\beta_r})$, $\chi = \mathcal{U}(\mathbb{S}_{\beta})$, and $\chi_1 = \mathcal{U}(\mathbb{S}_{\beta_1})$. Using the central limit theorem Gaussian approximation for the distribution of the coordinates of the term $\mathbf{R}\mathbf{e}_s$ in (49), their standard deviation is given by $\sigma\sigma_r\sqrt{(m + \ell)d}$, where $\sigma := \sqrt{\frac{1}{12}((2\beta + 1)^2 - 1)}$ and $\sigma_r := \sqrt{\frac{1}{12}((2\beta_r + 1)^2 - 1)}$ are the standard deviations of the $\mathbf{R}$ and $\mathbf{e}_s$ coordinates, respectively. Using a Gaussian tail bound and the upper bound $\beta_1$ for the coordinates of $\mathbf{e}'_s$, we have the approximate upper bound
$$B_f(\kappa, d) \leq \beta_1 + \sigma\sigma_r\sqrt{(m + \ell)d \cdot 2\ln(2)(\kappa + 2 + \log(hd))}. \tag{27}$$
- The existing correctness definition in Definition 2 assumes only 1 key generation run and 1 OPRF protocol evaluation. For a modified correctness definition with $Q_{\mathsf{eval}}$ total key generation and protocol evaluation pairs, the bound in (24) will be multiplied by $Q_{\mathsf{eval}}$.

In our evaluation protocol, the client obtains $\mathbf{u}_x - \mathbf{R}\mathbf{v}_k = \mathbf{B}_{x,t}\mathbf{k} + \mathbf{e}_f \in R_q^h$, where $\mathbf{e}_f = \mathbf{e}'_s - \mathbf{R}\mathbf{e}_s$ is an error term due to the server and client's randomness, and then rounds the result to the nearest multiple of $q/p$. The correctness proof bounds the probability of a PRF rounding error $\Pr[\lfloor \mathbf{B}_{x,t}\mathbf{k} + \mathbf{e}_f \rceil_p \neq \lfloor \mathbf{B}_{x,t}\mathbf{k} \rceil_p]$, which occurs only if $\mathbf{B}_{x,t}\mathbf{k}$ falls "close" to a rounding interval boundary (midpoints between consecutive multiples of $q/p$), i.e. within distance $B_f$ of an upper bound on the coordinates of error term $\mathbf{e}_f$. For this, we exploit the uniformity, up to negligible statistical distance $\epsilon_u$, of coordinates of $\mathbf{B}_{x,t}\mathbf{k}$ (a property of $\mathsf{H}$ which we call $\epsilon_u$-uniformity) and choose $q/p$ sufficiently large by a factor $\approx 2^\kappa$ compared to $B_f$ to achieve $2^{-\kappa}$ correctness error. We upper bound the $\epsilon_u$-uniformity of the BLMR13 and BP14 PRF instantiations of $\mathsf{H}$ for this purpose.

**Security analysis.** We first prove pseudorandomness against malicious clients with Theorem 5 and then request privacy against malicious servers (POPRIV2) with Theorem 6. Due to space constraints, we give only the sketch of the main proof steps, and refer the reader to Appendix D.3 and Appendix D.4 for the full proofs, respectively. In Appendix E.2, we discuss an issue in a proof of [TCR+22], where they prove that correctness and POPRIV2 together implies uniqueness (a stronger form of verifiability). We show that this implication also requires a *key binding* property, and for completeness, we provide a full proof of the statement that correctness, POPRIV2 and key binding implies uniqueness. Our LeOPaRd proposal satisfies all these properties. Although our correctness error is not necessarily less than $2^{-128}$, the verifiability property is not affected by the correctness error. We may only have a case where a different PRF output is computed in Finalize with probability $2^{-\kappa}$.

**Theorem 5.** *The POPRF construction* $\mathsf{F}$ *from Figures 8 and 9 satisfies pseudorandomness given in Definition 3, with random oracles* $\mathsf{RO}_r$ *and* $\mathsf{RO}_z$*, if:*

- *The client argument system* $\mathsf{NIZK}_1$ *is computationally sound and the server argument system* $\mathsf{NIZK}_2$ *is computationally zero-knowledge (Definition 18),*
- *The commitment scheme* $\mathsf{COM}$ *is computationally hiding and extractable (Definitions 15 and 17), and*
- *The* $\mathsf{MLWE\text{-}PRF\text{-}RU}_{\mathsf{param}}$ *assumption (Definition 10) holds for* $\mathsf{param} \coloneqq (\mathsf{H}, Q, Q^{\infty}_{x,t}, Q_{\mathsf{M}}, q, m, \ell+m, h, L, \beta_r, \overrightarrow{\chi})$ *and* $\overrightarrow{\chi} \coloneqq (\chi, \chi_1, \chi_k)$*, where* $Q$ *corresponds to the number of* $\mathsf{BlindEval}$ *queries,* $Q^{\infty}_{x,t}$ *to the maximum number of identical* $(x||t)$ *queries to* $\mathsf{BlindEval}$*, and* $Q_{\mathsf{M}}$ *to the number of* $\mathsf{RO}_r$ *queries, respectively.*

*More precisely, for any* PPT *adversary* $\mathcal{A}$*, there exist* PPT *adversaries* $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4, \mathcal{B}_5$ *and* $\mathcal{B}_6$ *against the computational zero-knowledge of* $\mathsf{NIZK}_2$*, computational soundness of* $\mathsf{NIZK}_1$*, hiding and extractability of* $\mathsf{COM}$*, and* $\mathsf{MLWE\text{-}PRF\text{-}RU}_{\mathsf{param}}$ *assumption, respectively, such that*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{po\text{-}prf}}_{\mathsf{F},\mathcal{A},\mathcal{S},\mathsf{RO}_r,\mathsf{RO}_z}(\lambda) \leq\ & \mathsf{Adv}^{\mathsf{CZK}}_{\mathsf{NIZK}_2,\mathcal{B}_1}(\lambda) + Q_{\mathsf{B}} \cdot \mathsf{Adv}^{\mathsf{CS}}_{\mathsf{NIZK}_1,\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathsf{CH}}_{\mathsf{COM},\mathcal{B}_3}(\lambda) \\ & + Q_{\mathsf{RO}_r} \cdot \mathsf{Adv}^{\mathsf{Ext}}_{\mathsf{COM},\mathcal{B}_4}(\lambda) + Q_{\mathsf{B}} \cdot \mathsf{Adv}^{\mathsf{Ext}}_{\mathsf{COM},\mathcal{B}_5}(\lambda) + \mathsf{Adv}^{\mathsf{iMLWE}_{\mathsf{param}}}_{\mathcal{B}_6}(\lambda) \\ & + Q_{\mathsf{RO}_z}\left(\frac{1}{p}+\frac{1}{q}\right)^{dh},\end{aligned}$$

*where* $Q_{\mathsf{B}}$*,* $Q_{\mathsf{RO}_r}$ *and* $Q_{\mathsf{RO}_z}$ *denote the number of* $\mathsf{BlindEval}$*,* $\mathsf{RO}_r$ *and* $\mathsf{RO}_z$ *queries, respectively, that the adversary* $\mathcal{A}$ *makes.*

Our proof essentially revolves around the idea of removing traces of the server's PRF key $\mathbf{k}$ from the computation of the POPRF output, such that the simulator $\mathcal{S}$ of the pseudorandomness game does not need any secret key dependent information. In order to do so, we first simulate the server proof $\pi_s$, such that from that point onward we do not need $\mathbf{k}$ for the generation of the proof. Next, we commit to an all zero vector $\mathbf{0}_m$ during the computation of the public key $\mathsf{pk}$, which allows us to remove $\mathbf{k}$ from the computation of $\mathsf{pk}$. Lastly, we rely on our $\mathsf{MLWE\text{-}PRF\text{-}RU}$ assumption to remove all traces of $\mathbf{k}$ from the computation of $\mathbf{u}_x$ and $\mathbf{v}_k$, and yet maintain the correctness of the scheme. Before relying on the $\mathsf{MLWE\text{-}PRF\text{-}RU}$ assumption though, we use the extractability property of the commitment scheme to extract the client's input $x$ and randomness $\mathbf{R}$, which we need for making the appropriate calls to the oracles of the $\mathsf{MLWE\text{-}PRF\text{-}RU}$ assumption. After applying all these changes, we end up in a position where we do not require the secret key $\mathbf{k}$ anymore, and hence, the simulator $\mathcal{S}$ can be constructed easily. We note that to make these argument go through, we also need to carefully program the random oracles $\mathsf{RO}_r$ and $\mathsf{RO}_z$ during the proof. Furthermore, our security proof takes into account the possibility of the adversary making many queries to various oracles, which are captured via the terms like $Q_{\mathsf{B}}$, $Q_{\mathsf{RO}_r}$, etc., and we ensure the hardness of the underlying problems under (polynomially-bounded) many queries.

**Theorem 6.** *The POPRF construction* $\mathsf{F}$ *from Figures 8 and 9 satisfies the request privacy against malicious servers (POPRIV2), given in Definition 4, with random oracles* $\mathsf{RO}_r$ *and* $\mathsf{RO}_z$*, if:*

- *The client argument system* $\mathsf{NIZK}_1$ *is computationally zero-knowledge and the server argument system* $\mathsf{NIZK}_2$ *is computationally sound (Definition 18),*

– *The commitment scheme* $\mathsf{COM}$ *is computationally hiding and extractable (Definitions 15 and 17), and*
– *The* $\mathsf{knMLWE}_{\ell+m,m,h,\chi}$ *assumption holds (Definition 7).*

*More precisely, for any* PPT *adversary* $\mathcal{A}$*, there exist* PPT *adversaries* $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$*,* $\mathcal{B}_4$*,* $\mathcal{B}_5$ *and* $\mathcal{B}_6$ *against the computational zero-knowledge of* $\mathsf{NIZK}_1$*, computational hiding of* $\mathsf{COM}$*, computational soundness of* $\mathsf{NIZK}_2$*, extractability of* $\mathsf{COM}$ *and* $\mathsf{knMLWE}_{m,\ell+m,h,\chi}$ *assumption, respectively, such that*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{po\text{-}priv\text{-}2}}_{\mathsf{F},\mathcal{A},\mathsf{RO}_r,\mathsf{RO}_z}(\lambda) \leq\ & \mathsf{Adv}^{\mathsf{CZK}}_{\mathsf{NIZK}_1,\mathcal{B}_1}(\lambda) + T \cdot Q_{\mathsf{P}} \cdot \mathsf{Adv}^{\mathsf{CH}}_{\mathsf{COM},\mathcal{B}_2}(\lambda) + 2Q_{\mathsf{R}} \cdot \mathsf{Adv}^{\mathsf{CH}}_{\mathsf{COM},\mathcal{B}_3}(\lambda) \\ & + 2Q_{\mathsf{F}} \cdot \mathsf{Adv}^{\mathsf{CS}}_{\mathsf{NIZK}_2,\mathcal{B}_4}(\lambda) + Q^{\mathsf{pk}}_{\mathsf{R}} \cdot \mathsf{Adv}^{\mathsf{Ext}}_{\mathsf{COM},\mathcal{B}_5}(\lambda) + 2Q_{\mathsf{R}} \cdot \mathsf{Adv}^{\mathsf{knMLWE}}_{\mathcal{B}_6}(\lambda)\,,\end{aligned}$$

*where* $Q_{\mathsf{P}}, Q_{\mathsf{R}}$ *and* $Q_{\mathsf{F}}$ *denote the number of* $\mathsf{PreProcClient}, \mathsf{Request}$ *and* $\mathsf{Finalize}$ *oracle queries, respectively, and* $Q^{\mathsf{pk}}_{\mathsf{R}}$ *denotes the number of queries to the* $\mathsf{Request}$ *oracle with distinct* $\mathsf{pk}$ *inputs that the adversary* $\mathcal{A}$ *makes.*

In order to prove request privacy against malicious servers, we aim to remove traces of the client's input $x$, such that at the end the transcript observed by the adversary is independent of the challenge bit $b$. To this end, we first simulate the client proof $\pi_c$, such that we do not require anymore the input $x$ for computing the proof. Next, we change the client commitments $\mathbf{c}_r$ and $\mathbf{d}_x$) to be commitments of all zero values $\mathbf{0}_{h\cdot(\ell+m)}$ and $0^{|x|}$ instead of $\mathbf{R}\ x$, respectively, which removes another occurrence of the input $x$. Finally, we rely on the Knapsack MLWE (Definition 7) assumption to replace computation of $\mathbf{C}_x$ with a uniformly random matrix. We note that in order to maintain the correctness of the scheme, during this last change we rely on the extractability of the commitment scheme. More precisely, we extract the secret key $\mathbf{k}$ from the public key $\mathsf{pk}$, and use $\mathbf{k}$ to correct the POPRF evaluation inside $\mathsf{Finalize}$. Importantly, we can use Regev-style [Reg05] *lightweight* commitments, as opposed to requiring *heavy* full trapdoors to extract from a commitment of the form $\mathbf{Ak} + \mathbf{e}$ used in [ADDS21].

At this point we have removed all occurrences of the input $x$, and hence, the adversary's view is independent of the input bit $b$.

## 5 Practical Performance Analysis

Our performance analysis focuses on estimating the sizes of various components of $\mathsf{LeOPaRd}$. Once protocol components are small enough, a practically-acceptable computational performance is often also achieved given that lattice-based schemes involve quite simple operations like matrix-vector multiplications (see examples in [ESLL19, EZS+19, LNS20, NS24]).

One of the most important factors impacting the performance of $\mathsf{LeOPaRd}$ is the size of the modulus $q$. Since the dimension parameters for suitable $\mathsf{MLWE}$ security grow linearly with $\log q$, the overall communication is in fact *quadratic* in $\log q$. Particularly, the sizes of commitments/encryptions (i.e, $(\mathbf{c}_r, \mathbf{C}_x)$ on the client side and $\mathbf{v}_k$ on the server side for $\mathsf{LeOPaRd}$) quickly become large for very large $q$. This is not just specific to our scheme, but true in general for lattice-based commitment/encryption schemes (as also used e.g. in [ADDS21]). Therefore, our first goal is to minimize $q$ as much as possible.

As shown in the correctness analysis (Lemma 7), the size of $q$ itself is heavily dominated by the $2^\kappa$ term. Therefore, given this bound is *statistical* and for correctness (rather than a security property), we consider a range of $\kappa$ values between 16 and 64. One may wonder what if a (malicious) server wants to stop the client from getting the correct OPRF result. There are two parts to consider here. First, as discussed in Remark 8, our correctness analysis easily extends to cover such a malicious server setting as the NIZK proof $\pi_s$ proves shortness of the server's error terms, and also we can simply increase $\kappa$. Second, OPRFs are often used in the server-client setting, and if clients do not receive the correct PRF output and therefore end up not receiving the intended service from the system, then the server would harm its own reputation and lose clients (while still not able to break client's privacy). Hence, we believe a rational server would not intentionally try to stop the client from getting the correct OPRF result.

For performance analysis, we focus on the instantiation of the message mapping obtained by truncating $\mathsf{H}^{\mathsf{bp}}$ of the BP14 PRF (see Appendix C.2) since this choice leads to a better performance overall. As discussed

in Appendix C, one may consider a general base $\gamma$ for the gadget matrix $\mathbf{G}$, i.e., $\mathbf{g} = (1, \gamma, \gamma^2, \ldots)$, and $\mathbf{G} = \mathbf{g} \otimes \mathbf{I}$.

We set $\chi = \mathcal{D}_s$, $\chi_1 = \mathcal{D}_{s_1}$ (matching our MLWE-PRF-RU analyses in Section 3) for width parameters $s, s_1$. Since $\chi_r$ (client's randomness distribution) is not involved in MLWE-PRF-RU analyses at all and the client's randomness $\mathbf{R}$ is one-time, we simply set $\chi_r = \mathcal{U}(\mathbb{S}_{\beta_r})$ for $\beta_r = 1$. Since BP14 PRF allows for a small secret key, we sample the server's key $\mathbf{k}$ from the same $\chi$ distribution, i.e., $\chi_k = \chi = \mathcal{D}_s$. To estimate the practical hardness of MLWE-PRF-RU problem, we rely on the reductions in Section 3, particularly the conditions in Theorem 2. Based on this theorem, we require the hardness of MLWE with the same secret dimension $m$ as in MLWE-PRF-RU and error Gaussian parameter $s_0$.

Our parameter setting then proceeds as follows. Fix a value for $\kappa \in \{16, 32, 64\}$. We iterate over different values of $d \in \{64, 128, 256\}$ that optimize the communication efficiency for both the server and the client. Recall that for BP14 PRF, we have $h = 1$. Therefore, for a fixed $d$, we also set $p$ as the smallest integer satisfying $(1/p+1/q)^{dh} \leq 2^{-128}$ (to satisfy a requirement from the pseudorandomness analysis in Theorem 5). Here, the term $1/q$ is much smaller (since $q$ is always larger than $2^{32}$) and therefore does not really play an important role. Then, based on Theorem 2, we first fix $s_B = s$, $s_{0,B} = s_0$ (choosing $\chi_{0,B}$ as spherical Gaussian), $\delta = \eta = 1$, $Q = 2^{64}$ and $\epsilon = 2^{-128}$, and compute the smallest $(s_0, s, \bar{s}_1, s_{1,B})$ values. The parameter $s_1 = \sqrt{s_{1,B}^2 + \bar{s}_1^2}$ is set as in Theorem 1. Here we set $Q_x^\infty = 2^\kappa$ for the fully oblivious setting, and $Q_x^\infty = 2^{16}$ for the partially oblivious setting. We explain the reasoning behind this towards the end of this section. In this procedure, we have $N = m + \ell$ and we estimate the values for $m$ and $\ell$ initially, and correct the estimate iteratively for accuracy.

Now we consider the size of $q$ based on the correctness analysis (Lemma 7) and set $\log q$ as small as possible. Then, we set the smallest value for $m$ so that MLWE security with error width $s_0$ at 128 bits is achieved (server security). As discussed above, this comes from the security reductions in Section 3. Similarly, we set the smallest value for $\ell$ so that $\mathsf{MLWE}_{\ell,\chi_r}$ security at 128 bits is achieved (client security). We measure the practical hardness of MLWE using the lattice estimator [APS15] and aim for a "root Hermite factor" (RHF) of around 1.0045 as in earlier works [ESLR23, ESZ22b, LNP22, ESLL19]. RHF is a commonly used measure to estimate the quality of lattice reduction to solve a particular lattice problem. Earlier works such as [ESLR23, ESZ22b, LNP22, ESLL19] considered the same RHF value for 128-bit security level. The same MLWE dimension parameters $w_c = \ell$ (by client) and $w_s = m$ (by server) can be used to establish the hiding property of commitment scheme used by the client and the server as they have the same $\log q$ and their randomnesses are sampled from the same distributions as the earlier MLWE error terms. We also note that given many MLWE and MLWE-PRF-RU samples may be leaked due to many OPRF evaluation queries, we also need to take into account combinatorial attacks (e.g., using Gröbner basis) against MLWE. We verified using the lattice estimator [APS15] that the combinatorial attack complexity is always above $2^{128}$ for our parameter settings.

Once the majority of the parameters are selected as above, what remains is to consider (i) the binding property of the commitment scheme, (ii) the indistinguishability of a commitment key with a trapdoor from a regular commitment key, and (iii) decryption/extraction correctness for the commitments. First we note that the $\mathbf{d}_x$ commitment simply commits to a small input $x$ with entropy at most 256 bits (probably even less). Therefore, we can simply use a regular lattice-based encryption for $\mathbf{d}_x$, which we assume to cost 1 KB. Now for the other commitments, given that we are dealing with (relatively) large moduli, we see that the third requirement above is easily met. Again due to (relatively) large moduli and that the MSIS solution norm bounds are quite small (due to tight norm-bound NIZK proofs), the second requirement turns out to be more dominant than the first one in setting the values for $(n_s, n_c)$ (the $n$ parameters from Appendix F used by the server and client, respectively). As discussed in Appendix F, we require the same MLWE assumptions over $R_q$ with error width $s_0$ (for server's commitment) and errors sampled from $\chi_r$ (for client's commitment). As a result, we get that $n_s = m$ and $n_c = \ell$. Note that the dimensions of the randomnesses used inside the commitments by the server and the client do not have a significant impact in our parameter setting and efficiency estimates since the well-formedness of the commitments are proven by LaBRADOR, and therefore, this proof cost gets amortized along with everything else (see below for more discussion on the proof costs). For the MSIS hardness of our parameter settings, it turns out that module ranks (from both the client and

the server's side) as small as 1-2 is already sufficient. Therefore, we can comfortably use the standard low-order bit-dropping technique from [DKL+18]. For simplicity, we assume $D = 12$ bits can be dropped in the 'binding' parts of the commitments $\mathbf{c}_r$ and $\mathbf{c}_k$ (i.e., top $n_c$ and $n_s$ rows of $\mathbf{c}_r$ and $\mathbf{c}_k$, respectively). A similar $D$ parameter has been used in earlier works with smaller moduli such as [ESLR23, LNP22]. We believe that a larger $D$ can be used, but the saving is not very significant, and therefore, we opt to not pursue further analysis here so as not to over-complicate the discussions. This concludes the setting of all parameters for LeOPaRd except the underlying NIZK proofs.

In Table 3, we present some example parameter settings and communication sizes aiming at 128-bit security level. The communication of the server's public key is a *one-time* offline cost. Here, we consider two cases: (i) fully oblivious (i.e. no tag) setting where $Q_x^\infty = 2^\kappa$; and (ii) partially oblivious setting where $Q_{x,t}^\infty = 2^{16}$ (both corresponding to $Q_x^\infty$ in Theorem 2). In the former case, we are restricting the total number of OPRF queries to $2^\kappa$, matching also the correctness error. In the latter case, we consider that the server has the ability to see a part of the PRF input, i.e. tag $t$, and therefore, can limit the number of OPRF queries under the same tag to be at most $2^{16}$. If the tag corresponds to a user identifier as in [ECS+15], then this would mean $2^{16}$ queries per identifier, which we believe is quite reasonable (for honest users) in practice. We also note that our parameter setting aiming at 90-95 bits of security (to compare with [AG24] in Table 1) is provided in Table 4.

| Scheme | $\kappa$ | $m = n_s$ | $\ell = n_c$ | $s_0$ | $s$ | $s_1$ | $\log q$ | Server PK size | Server Offline (per query) | Server Online (w/o NIZK) | Client Offline (per query) | Client Online (w/o NIZK) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LeOPaRd | 16 | 24 | 27 | 9.90 | 21.5 | 11262 | 42 | 13.50 | 16.73 | 0.33 | 23.39 | 8.88 |
| Partial | 32 | 34 | 37 | 9.90 | 21.6 | 12866 | 59 | 28.16 | 32.73 | 0.46 | 46.77 | 16.67 |
| OPRF | 64 | 54 | 56 | 9.93 | 21.6 | 15535 | 92 | 72.56 | 79.06 | 0.72 | 114.78 | 39.81 |
| LeOPaRd | 16 | 24 | 27 | 9.90 | 21.5 | 11262 | 42 | 13.50 | 16.73 | 0.33 | 23.39 | 8.88 |
| (plain) | 32 | 38 | 41 | 10.25 | 23.5 | $\approx 2^{21}$ | 66 | 35.62 | 40.73 | 0.52 | 58.55 | 20.59 |
| OPRF | 64 | 67 | 70 | 10.93 | 28.5 | $\approx 2^{37}$ | 114 | 113.06 | 122.02 | 0.89 | 178.69 | 60.67 |

Table 3: Summary of our parameters aiming around 128 bit security and communication cost results. All communication and PK sizes are in KB. For all settings, we have $d = 64$, $p = 4$, $h = 1$, and $\beta_r = 1$.

Now we get to estimating the sizes of LaBRADOR-based [BS23] NIZKs (in combination with LNP22 [LNP22] to achieve zero-knowledge). One can observe that its proof size is barely impacted by the witness size (see, e.g., [BS23, Fig. 1]) thanks to its recursive nature. In [ADDG24], the authors report that their LaBRADOR-based well-formedness NIZK proof (including the LNP22 part) under a 75-bit modulus for FHE ciphertexts is about 45 KB for a single query. We believe the proofs needed for LeOPaRd more naturally fit LaBRADOR because, unlike [ADDG24], we do not need to use a different *proof* modulus than the one used in the core OPRF protocol. Hence, we estimate our proof cost to be about 45 KB (for a single query). Our estimate is also inline with similar LaBRADOR-based proof estimates in [AG24]. In fact, our estimates may even be more conservative than [AG24] because their modulus $q$ is much larger around $2^{143}$-$2^{169}$. In Table 3, we exclude the costs of NIZKs to provide a clearer presentation of the advantage of our core techniques (that are outside of the NIZKs). From here, we can see that reducing the costs of underlying NIZKs is a promising step to improve the efficiency of post-quantum OPRFs, which we leave as a future work.

## Acknowledgements

We want to thank the anonymous reviewers for their very helpful comments and suggestions. We want to thank Martin Albrecht for helpful discussions and for answering our questions regarding the prior OPRF works. This work was (partially) supported by an Amazon Research Award Fall 2023, the France 2030 ANR Project ANR-22-PECY-003 SecureCompute, by the Austrian Science Fund (FWF) Project J4879-N, by the Bakar Funds and Peder Sather Funds, and Australian Research Council Discovery Grants DP180102199 and

| Scheme | $\kappa$ | $m = n_s$ | $\ell = n_c$ | $s_0$ | $s$ | $s_1$ | $\log q$ | Server PK size | Server Offline (per query) | Server Online (w/o NIZK) | Client Offline (per query) | Client Online (w/o NIZK) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LeOPaRd | 16 | 18 | 20 | 9.88 | 21.4 | 10088 | 42 | 10.12 | 12.47 | 0.33 | 17.48 | 6.91 |
| Partial | 32 | 25 | 27 | 9.90 | 21.5 | 11347 | 59 | 20.70 | 23.97 | 0.46 | 34.34 | 12.52 |
| OPRF | 64 | 40 | 42 | 9.93 | 21.6 | 13670 | 91 | 53.12 | 58.30 | 0.71 | 84.93 | 29.44 |
| LeOPaRd | 16 | 18 | 20 | 9.88 | 21.4 | 10088 | 42 | 10.12 | 12.47 | 0.33 | 17.48 | 6.91 |
| (plain) | 32 | 28 | 30 | 10.25 | 23.5 | $\approx 2^{21}$ | 66 | 26.25 | 29.91 | 0.52 | 43.08 | 15.44 |
| OPRF | 64 | 50 | 52 | 10.93 | 28.4 | $\approx 2^{37}$ | 114 | 84.38 | 90.84 | 0.89 | 133.17 | 45.53 |
| [AG24] | $Q = 2^{16}$; 95-bit security | | | | | | 143 | 108.30 | N/A | 1.80 | N/A | 71.50 |
| (plain) | $Q = 2^{32}$; 90-bit security | | | | | | 151 | 113.50 | N/A | 1.80 | N/A | 75.50 |
| OPRF | $Q = 2^{64}$; 167-bit security | | | | | | 169 | 221.50 | N/A | 1.80 | N/A | 169.00 |

Table 4: Summary of our parameters aiming around 90-95 bit security and communication cost results. All communication and PK sizes are in KB. For all settings, we have $d = 64$, $p = 4$, $h = 1$, and $\beta_r = 1$. We also provide the results for [AG24] as the most closely-related work. $Q$ denotes a global upperbound on the number of evaluation queries (as in Table 1).

DP220101234. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of Amazon or other funding agencies.

## References

ADDG24. Martin R. Albrecht, Alex Davidson, Amit Deo, and Daniel Gardham. Crypto dark matter on the torus - oblivious PRFs from shallow PRFs and TFHE. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VI*, volume 14656 of *LNCS*, pages 447–476. Springer, Cham, May 2024. 2, 3, 8, 26, 30, 65

ADDS21. Martin R. Albrecht, Alex Davidson, Amit Deo, and Nigel P. Smart. Round-optimal verifiable oblivious pseudorandom functions from ideal lattices. In Juan Garay, editor, *PKC 2021, Part II*, volume 12711 of *LNCS*, pages 261–289. Springer, Cham, May 2021. 2, 3, 4, 5, 7, 22, 28, 37, 39, 65, 66

AG24. Martin R. Albrecht and Kamil Doruk Gür. Verifiable oblivious pseudorandom functions from lattices: Practical-ish and thresholdisable. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part IV*, volume 15487 of *LNCS*, pages 205–237. Springer, Singapore, December 2024. 2, 3, 4, 7, 25, 30, 31, 37, 38

ALS20. Thomas Attema, Vadim Lyubashevsky, and Gregor Seiler. Practical product proofs for lattice commitments. In *CRYPTO (2)*, LNCS, pages 470–499. Springer, 2020. 37

APRR24. Navid Alamati, Guru-Vamsi Policharla, Srinivasan Raghuraman, and Peter Rindal. Improved alternating-moduli PRFs and post-quantum signatures. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VIII*, volume 14927 of *LNCS*, pages 274–308. Springer, Cham, August 2024. 24

APS15. Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *J. Math. Cryptol.*, 9(3):169–203, 2015. 29

BDFH25. Ward Beullens, Lucas Dodgson, Sebastian H. Faller, and Julia Hesse. The 2Hash OPRF framework and efficient post-quantum instantiations. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part VIII*, volume 15608 of *LNCS*, pages 332–362. Springer, Cham, May 2025. 2, 3, 24

BDL+18. Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. More efficient commitments from structured lattice assumptions. In *SCN*, volume 11035 of *Lecture Notes in Computer Science*, pages 368–385. Springer, 2018. 23, 25, 69

BFM88. Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th ACM STOC*, pages 103–112. ACM Press, May 1988. 36

BIP+18. Dan Boneh, Yuval Ishai, Alain Passelègue, Amit Sahai, and David J. Wu. Exploring crypto dark matter: New simple PRF candidates and their applications. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 699–729. Springer, Cham, November 2018. 3

BLMR13. Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic prfs and their applications. In *CRYPTO (1)*, volume 8042 of *LNCS*, pages 410–428. Springer, 2013. 6, 13, 21, 26, 38, 68

BLNS23. Ward Beullens, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Lattice-based blind signatures: Short, efficient, and round-optimal. In *CCS*, pages 16–29. ACM, 2023. 39

BLP+13. Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *STOC*, pages 575–584. ACM, 2013. 11

Bol03. Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 31–46. Springer, Berlin, Heidelberg, January 2003. 66

BP14. Abhishek Banerjee and Chris Peikert. New and improved key-homomorphic pseudorandom functions. In *CRYPTO (1)*, volume 8616 of *LNCS*, pages 353–370. Springer, 2014. 6, 12, 13, 21, 22, 26, 38, 41

BS23. Ward Beullens and Gregor Seiler. Labrador: Compact proofs for R1CS from module-sis. In *CRYPTO (5)*, volume 14085 of *LNCS*, pages 518–548. Springer, 2023. 2, 4, 26, 30, 37, 40

CHL22. Sílvia Casacuberta, Julia Hesse, and Anja Lehmann. SoK: Oblivious pseudorandom functions. In *2022 IEEE European Symposium on Security and Privacy*, pages 625–646. IEEE Computer Society Press, June 2022. 1

CJ25. Jung Hee Cheon and Daehyun Jang. Cryptanalysis on lightweight verifiable homomorphic encryption. In *ASIACRYPT (7)*, volume 16251 of *Lecture Notes in Computer Science*, pages 366–397. Springer, 2025. 3

COS20. Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. In *EUROCRYPT (1)*, volume 12105 of *Lecture Notes in Computer Science*, pages 769–793. Springer, 2020. 39

DDT25. Alex Davidson, Amit Deo, and Louis Tremblay Thibault. Pool: A practical OT-based OPRF from learning with rounding. In Chun-Ying Huang, Jyh-Cheng Chen, Shiuh-Pyng Shieh, David Lie, and Véronique Cortier, editors, *ACM CCS 2025*, pages 1038–1052. ACM Press, October 2025. 3, 4, 8, 10, 24

DGS+18. Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. Privacy pass: Bypassing internet challenges anonymously. *PoPETs*, 2018(3):164–180, July 2018. 1, 25

Di 03. Giovanni Di Crescenzo. Equivocable and extractable commitment schemes. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02*, volume 2576 of *LNCS*, pages 74–87. Springer, Berlin, Heidelberg, September 2003. 35

DKL+18. Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):238–268, 2018. 30

DPS23. Julien Devevey, Alain Passelègue, and Damien Stehlé. G+G: A fiat-shamir lattice signature based on convolved gaussians. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part VII*, volume 14444 of *LNCS*, pages 37–64. Springer, Singapore, December 2023. 11

ECS+15. Adam Everspaugh, Rahul Chatterjee, Samuel Scott, Ari Juels, and Thomas Ristenpart. The Pythia PRF service. In *USENIX Security Symposium*, pages 547–562. USENIX Association, 2015. 30

ENP24. Thomas Espitau, Guilhem Niot, and Thomas Prest. Flood and submerse: Distributed key generation and robust threshold signature from lattices. In *CRYPTO (7)*, volume 14926 of *Lecture Notes in Computer Science*, pages 425–458. Springer, 2024. 6, 7, 11, 13, 16, 17

ENS20. Muhammed F. Esgin, Ngoc Khanh Nguyen, and Gregor Seiler. Practical exact proofs from lattices: New techniques to exploit fully-splitting rings. In *ASIACRYPT (2)*, volume 12492 of *LNCS*, pages 259–288. Springer, 2020. 37

ESLL19. Muhammed F. Esgin, Ron Steinfeld, Joseph K. Liu, and Dongxi Liu. Lattice-based zero-knowledge proofs: New techniques for shorter and faster constructions and applications. In *CRYPTO (1)*, volume 11692 of *Lecture Notes in Computer Science*, pages 115–146. Springer, 2019. (Full version at `ia.cr/2019/445`). 28, 29

ESLR23. Muhammed F. Esgin, Ron Steinfeld, Dongxi Liu, and Sushmita Ruj. Efficient hybrid exact/relaxed lattice proofs and applications to rounding and VRFs. In *CRYPTO (5)*, volume 14085 of *LNCS*, pages 484–517. Springer, 2023. (Full version at `ia.cr/2022/141`). 29, 30, 37

ESZ22a. Muhammed F. Esgin, Ron Steinfeld, and Raymond K. Zhao. Efficient verifiable partially-decryptable commitments from lattices and applications. In *Public Key Cryptography (PKC) (1)*, volume 13177 of *Lecture Notes in Computer Science*, pages 317–348. Springer, 2022. (Full version at `ia.cr/2022/142`). 69

ESZ22b. Muhammed F. Esgin, Ron Steinfeld, and Raymond K. Zhao. MatRiCT+: More efficient post-quantum private blockchain payments. In *IEEE Symposium on Security and Privacy (S&P)*, pages 1281–1298. IEEE, 2022. (Full version at `ia.cr/2021/545`). 29

EZS+19. Muhammed F. Esgin, Raymond K. Zhao, Ron Steinfeld, Joseph K. Liu, and Dongxi Liu. MatRiCT: Efficient, scalable and post-quantum blockchain confidential transactions protocol. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 567–584. ACM, 2019. (Full version at ia.cr/2019/1287). 11, 28

FIPR05. Michael J. Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword search and oblivious pseudorandom functions. In Joe Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 303–324. Springer, Berlin, Heidelberg, February 2005. 2

GdKQ+24. Phillip Gajland, Bor de Kock, Miguel Quaresma, Giulio Malavolta, and Peter Schwabe. SWOOSH: Efficient lattice-based non-interactive key exchange. In Davide Balzarotti and Wenyuan Xu, editors, *USENIX Security 2024*. USENIX Association, August 2024. 4

GKR+21. Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. Poseidon: A new hash function for zero-knowledge proof systems. In *USENIX Security Symposium*, pages 519–535. USENIX Association, 2021. 39

GSW13. Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO (1)*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92. Springer, 2013. 68

Hei25. Lena Heimberger. Private communication, 2025. 4

HKL+25. Lena Heimberger, Daniel Kales, Riccardo Lolato, Omid Mir, Sebastian Ramacher, and Christian Rechberger. Leap: A fast, lattice-based OPRF with application to private set intersection. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part VII*, volume 15607 of *LNCS*, pages 254–283. Springer, Cham, May 2025. 3, 4, 24

JKK14. Stanislaw Jarecki, Aggelos Kiayias, and Hugo Krawczyk. Round-optimal password-protected secret sharing and T-PAKE in the password-only model. In *ASIACRYPT (2)*, volume 8874 of *Lecture Notes in Computer Science*, pages 233–253. Springer, 2014. 1, 2, 3, 4, 5, 22, 24

JKKX16. Stanislaw Jarecki, Aggelos Kiayias, Hugo Krawczyk, and Jiayu Xu. Highly-efficient and composable password-protected secret sharing (or: How to protect your Bitcoin wallet online). In *2016 IEEE European Symposium on Security and Privacy*, pages 276–291. IEEE Computer Society Press, March 2016. 1, 24

JKX18. Stanislaw Jarecki, Hugo Krawczyk, and Jiayu Xu. OPAQUE: An asymmetric PAKE protocol secure against pre-computation attacks. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 456–486. Springer, Cham, April / May 2018. 1, 4, 24

JL09. Stanislaw Jarecki and Xiaomin Liu. Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 577–594. Springer, Berlin, Heidelberg, March 2009. 2

Kat21. Shuichi Katsumata. A new simple technique to bootstrap various lattice zero-knowledge proofs to QROM secure NIZKs. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 580–610, Virtual Event, August 2021. Springer, Cham. 4, 65

KBR13. Sriram Keelveedhi, Mihir Bellare, and Thomas Ristenpart. DupLESS: Server-aided encryption for deduplicated storage. In Samuel T. King, editor, *USENIX Security 2013*, pages 179–194. USENIX Association, August 2013. 1

KLSS23. Duhyeong Kim, Dongwon Lee, Jinyeong Seo, and Yongsoo Song. Toward practical lattice-based proof of knowledge from hint-mlwe. In *CRYPTO (5)*, volume 14085 of *Lecture Notes in Computer Science*, pages 549–580. Springer, 2023. 6, 7, 11, 13, 16, 17

LNP22. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plançon. Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. In *CRYPTO (2)*, volume 13508 of *LNCS*, pages 71–101. Springer, 2022. Full version at (https://eprint.iacr.org/archive/2022/284/20220814:160144). 4, 25, 29, 30, 37, 69

LNPS21. Vadim Lyubashevsky, Ngoc Khanh Nguyen, Maxime Plançon, and Gregor Seiler. Shorter lattice-based group signatures via "almost free" encryption and other optimizations. In *ASIACRYPT (4)*, volume 13093 of *LNCS*, pages 218–248. Springer, 2021. 68, 69

LNS20. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Practical lattice-based zero-knowledge proofs for integer relations. In *ACM CCS*, pages 1051–1070. ACM, 2020. 28, 37

LPA+19. Lucy Li, Bijeeta Pal, Junade Ali, Nick Sullivan, Rahul Chatterjee, and Thomas Ristenpart. Protocols for checking compromised credentials. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 1387–1403. ACM Press, November 2019. 1

MR04. Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. In *FOCS*, pages 372–381. IEEE Computer Society, 2004. 11

NO25. Ngoc Khanh Nguyen and George O'Rourke. More efficient lattice-based zero-knowledge proofs with straight-line extractability. In *APKC@AsiaCCS*, pages 34–43. ACM, 2025. 4

NS24. Ngoc Khanh Nguyen and Gregor Seiler. Greyhound: Fast polynomial commitments from lattices. In *CRYPTO (10)*, volume 14929 of *LNCS*, pages 243–275. Springer, 2024. 28, 40

Pei10. Chris Peikert. An efficient and parallel gaussian sampler for lattices. In *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 80–97. Springer, 2010. 11

Reg05. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93. ACM, 2005. 23, 28, 68

SSS23. Shifeng Sun, Ron Steinfeld, and Amin Sakzad. Incremental symmetric puncturable encryption with support for unbounded number of punctures. *Des. Codes Cryptogr.*, 91(4):1401–1426, 2023. 38, 48

TCR+22. Nirvan Tyagi, Sofía Celi, Thomas Ristenpart, Nick Sullivan, Stefano Tessaro, and Christopher A. Wood. A fast and simple partially oblivious PRF, with applications. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 674–705. Springer, Cham, May / June 2022. 1, 2, 3, 4, 5, 8, 9, 10, 23, 27, 65, 66

TPY+19. Kurt Thomas, Jennifer Pullman, Kevin Yeo, Ananth Raghunathan, Patrick Gage Kelley, Luca Invernizzi, Borbala Benko, Tadek Pietraszek, Sarvar Patel, Dan Boneh, and Elie Bursztein. Protecting accounts from credential stuffing with password breach alerting. In Nadia Heninger and Patrick Traynor, editors, *USENIX Security 2019*, pages 1556–1571. USENIX Association, August 2019. 1

YAZ+19. Rupeng Yang, Man Ho Au, Zhenfei Zhang, Qiuliang Xu, Zuoxia Yu, and William Whyte. Efficient lattice-based zero-knowledge arguments with standard soundness: Construction and applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 147–175. Springer, Cham, August 2019. 65

YBH+25. Yibin Yang, Fabrice Benhamouda, Shai Halevi, Hugo Krawczyk, and Tal Rabin. Gold OPRF: Post-quantum oblivious power-residue PRF. In Marina Blanton, William Enck, and Cristina Nita-Rotaru, editors, *2025 IEEE Symposium on Security and Privacy*, pages 259–278. IEEE Computer Society Press, May 2025. 2, 3, 4

ZSE+24. Xinyu Zhang, Ron Steinfeld, Muhammed F. Esgin, Joseph K. Liu, Dongxi Liu, and Sushmita Ruj. Loquat: A snark-friendly post-quantum signature based on the legendre PRF with applications in ring and aggregate signatures. In *CRYPTO (1)*, volume 14920 of *LNCS*, pages 3–38. Springer, 2024. 39

# A Additional Preliminaries

## A.1 Extractable Commitment Scheme

In this work, we consider an extractable commitment scheme [Di 03], which is formally defined as follows.

**Definition 14 (Extractable Commitment Scheme).** *A (non-interactive) extractable commitment scheme consists of a tuple of algorithms* $\mathsf{COM} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Verify})$ *defined as follows:*

$\mathsf{Setup}(1^\lambda)$**:** *is a* PPT *algorithm that on input a (unary encoded) security parameter* $\lambda$*, outputs a commitment key* $\mathsf{ck}$*.*

$\mathsf{Commit}(\mathsf{ck}, m)$**:** *is a* PPT *algorithm that on input a commitment key* $\mathsf{ck}$ *and a message* $m \in \{0,1\}^\lambda$*, outputs a commitment* $c$ *and an opening information* $d$*.*

$\mathsf{Verify}(\mathsf{ck}, c, d, m)$**:** *is a* DPT *algorithm that on input a commitment key* $\mathsf{ck}$*, commitment* $c$*, an opening information* $d$ *and a message* $m \in \{0,1\}^\lambda$*, outputs a bit* $b \in \{0,1\}$*.*

In the above $\mathsf{Commit}$ definition, we leave the randomness part implicit and assume that it is generated internally inside the $\mathsf{Commit}$ function. When proving well-formedness of commitments via a zero-knowledge proof, the randomness will also be part of the prover's witness. When we need to specify the randomness $r$ in such cases, we will write $\mathsf{Commit}(\mathsf{ck}, m; r)$ to explicitly refer to the internally generated randomness of the commitment.

We require the standard notion of *correctness*, which says that for every $\lambda \in \mathbb{N}$, every $\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda)$ and every message $m \in \{0,1\}^\lambda$, it holds that

$$\Pr\left[\mathsf{Verify}(\mathsf{ck}, \mathsf{Commit}(\mathsf{ck}, m), m) = 1\right] = 1.$$

In terms of security, we require the commitment scheme to satisfy *computational hiding* and *computational binding* properties, along with *extractability*.

**Definition 15 (Computational Hiding).** *A commitment scheme* $\mathsf{COM}$ *is computationally hiding if for every* PPT *adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}$*, such that*

$$\Pr\left[\mathsf{CH}_{\mathsf{COM},\mathcal{A}}(\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

*where the experiment* $\mathsf{CH}_{\mathsf{COM},\mathcal{A}}$ *is defined as follows*

$\mathsf{CH}_{\mathsf{COM},\mathcal{A}}(\lambda)$

1 : $\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda)$
2 : $(m_0, m_1) \leftarrow \mathcal{A}(\mathsf{ck})$
3 : $b \xleftarrow{\$} \{0,1\}$
4 : $(c, d) \leftarrow \mathsf{Commit}(\mathsf{ck}, m_b)$
5 : $b' \leftarrow \mathcal{A}(c)$
6 : **return** $b = b'$

**Definition 16 (Computational Binding).** *A commitment scheme* $\mathsf{COM}$ *is computationally binding if for every* PPT *adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}$*, such that*

$$\Pr\left[\mathsf{CB}_{\mathsf{COM},\mathcal{A}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda),$$

*where the experiment* $\mathsf{CB}_{\mathsf{COM},\mathcal{A}}$ *is defined as follows*

$\mathsf{CB}_{\mathsf{COM},\mathcal{A}}(\lambda)$

1 : $\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda)$
2 : $(c, d, d', m, m') \leftarrow \mathcal{A}(\mathsf{ck})$
3 : **assert** $m \neq m'$
4 : **return** $\mathsf{Verify}(\mathsf{ck}, c, d, m) = 1 \wedge \mathsf{Verify}(\mathsf{ck}, c, d', m') = 1$

**Definition 17 (Extractability).** *A commitment scheme* COM *is extractable, if there exists a pair of* PPT *algorithm* $\mathcal{E} = (\mathcal{E}_1, \mathcal{E}_2)$*, called the extractor, such that the for all* $m \in \{0,1\}^\lambda$ *and for every* PPT *adversary* $\mathcal{A}$*, there exists a negligible function* negl*, such that*

$$\left| \Pr\left[\mathcal{A}(\mathsf{ck}) = 1 \mid \mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda)\right] - \Pr\left[\mathcal{A}(\mathsf{ck}) = 1 \mid (\mathsf{ck}, \mathsf{td}) \leftarrow \mathcal{E}_1(1^\lambda)\right] \right| \leq \mathsf{negl}(\lambda),$$

*and*

$$\Pr\left[\begin{array}{c|c} \mathsf{Verify}(c,d,m) = 1 & (\mathsf{ck}, \mathsf{td}) \leftarrow \mathcal{E}_1(1^\lambda) \\ \wedge & (c,d,m) \leftarrow \mathcal{A}(\mathsf{ck}) \\ m \neq m' & m' \leftarrow \mathcal{E}_2(\mathsf{ck}, \mathsf{td}, c) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

### A.2 Non-Interactive Zero-Knowledge Arguments

Let $R$ be a binary relation and $L$ the language consisting of statements in $R$. We formally define a non-interactive zero-knowledge (NIZK) argument system [BFM88] as follows.

**Definition 18 (Non-Interactive Zero-Knowledge Argument System).** *A non-interactive zero-knowledge (NIZK) argument system* NIZK *for a language* $L \in$ NP *(with witness relation* $R$*) is a tuple of algorithms* $\mathsf{NIZK} = (\mathsf{PGen}, \mathsf{P}, \mathsf{V})$*, such that:*

$\mathsf{PGen}(1^\lambda)$**:** *is a* PPT *algorithm that on input a (unary encoded) security parameter* $\lambda$*, outputs a common reference string* crs*.*

$\mathsf{P}(\mathsf{crs}, x, w)$**:** *is a* PPT *algorithm that on input a common reference string* crs*, a statement* $x$ *and a witness* $w$*, outputs a proof* $\pi$*.*

$\mathsf{V}(\mathsf{crs}, x, \pi)$**:** *is a* DPT *algorithm that on input a common reference string* crs*, a statement* $x$ *and a proof* $\pi$*, outputs a bit* $b$*.*

*We require* NIZK *to meet the following properties:*

***Perfect Completeness.*** *For every* $(x, w) \in R$ *we have that*

$$\Pr\left[\mathsf{V}(\mathsf{crs}, x, \pi) = 1 \mid \mathsf{crs} \leftarrow \mathsf{PGen}(1^\lambda), \pi \leftarrow \mathsf{P}(\mathsf{crs}, x, w)\right] = 1.$$

***Computational Soundness.*** *For every* $x \notin L$*, and every* PPT *adversary* $\mathcal{A}$*, we have that*

$$\Pr\left[\mathsf{V}(\mathsf{crs}, x, \pi) = 1 \mid \mathsf{crs} \leftarrow \mathsf{PGen}(1^\lambda), \pi \leftarrow \mathcal{A}(\mathsf{crs}, x)\right] \leq \mathsf{negl}(\lambda).$$

***Computational Zero-Knowledge.*** *There exists a* PPT *algorithm* $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ *such that for every* PPT *adversary* $\mathcal{A}$*,*

$$\begin{aligned} &\left| \Pr\left[\mathcal{A}^{\mathsf{P}(\mathsf{crs},\cdot,\cdot)}(\mathsf{crs}) = 1 \mid \mathsf{crs} \leftarrow \mathsf{PGen}(1^\lambda)\right]\right. \\ &\left. - \Pr\left[\mathcal{A}^{\mathcal{O}(\mathsf{crs}_s,\tau_s,\cdot,\cdot)}(\mathsf{crs}) = 1 \mid (\mathsf{crs}_s, \tau_s) \leftarrow \mathcal{S}_1(1^\lambda)\right] \right| \leq \mathsf{negl}(\lambda), \end{aligned}$$

*where* $\mathcal{O}(\mathsf{crs}_s, \tau_s, \cdot, \cdot)$ *is an oracle that outputs* $\perp$ *on input* $(x, w)$ *when* $(x, w) \notin R$ *and outputs* $\pi \leftarrow \mathcal{S}_2(\mathsf{crs}_s, \tau_s, x)$ *when* $(x, w) \in R$*.*

## B Averaging Attack Against MLWE-PRF-RU

In this section we describe a simple error "averaging" attack on MLWE-PRF-RU (and our OPRF protocol as well as those using the same blueprint [ADDS21, AG24]).

In this attack, the adversary uses the $Q_x^\infty$ samples on same $x$ queried to the oracle to compute $\mathbf{z}'_j := \mathbf{u}_j - \mathbf{R}_j\mathbf{c}_j = \mathbf{B}_x\mathbf{k} + \bar{\mathbf{e}}'_j - \mathbf{R}_j\mathbf{e}_j$ for $j \in [Q_x^\infty]$. The attacker then computes the average (over the rationals) of these noisy secrets to get $\bar{\mathbf{z}} := \frac{1}{Q_x^\infty}\sum_{j\in[Q_x^\infty]}\mathbf{z}'_j = \mathbf{B}_x\mathbf{k} + \bar{\mathbf{e}}$ in which the error $\bar{\mathbf{e}} := \frac{1}{Q_x^\infty}\sum_{j\in[Q_x^\infty]}(\bar{\mathbf{e}}'_j - \mathbf{R}_j\mathbf{e}_j)$ has a reduced standard deviation $\bar{\sigma} \approx \sigma_1/\sqrt{Q_x^\infty}$. If $\bar{\sigma}$ is a constant factor smaller than $1/2$, all the $\bar{\mathbf{e}}$ error coordinates will likely be $< 1/2$ and rounding $\bar{\mathbf{z}}$ to the nearest integers will give the attacker the secret $\mathbf{B}_x\mathbf{k}$ (and hence $\mathbf{k}$) with high probability. Consequently, to prevent the averaging attack, a necessary condition is $\sigma_1 \geq \sqrt{Q_x^\infty}/c$ for a small constant $c > 1$. More generally, a necessary condition to prevent the averaging attack from reducing the error standard deviation (and MLWE security) below the $\sigma$ standard deviation used to set MLWE security in the samples $\mathbf{c}_j$, is that $\sigma_1/\sigma \geq \sqrt{Q_x^\infty}/c$. This lower bound matches (up to the small factor $c$) our MLWE-PRF-RU security reduction parameter lower bound discussed above (for large $Q_x^\infty$), showing the optimality of our reduction condition up to a small factor. Note that the attack and our security reduction bounds only depend on the maximum no. of samples queried per $x$ value. This is in contrast to the suboptimal Rényi-divergence-based security analysis in [AG24], which seems to inherently lose a factor proportional to $\sqrt{Q}$, where $Q$ is the total number of oracle queries over *all* $x$ values.

## C Instantiating NIZK Proofs and Message Mapping H

In this section, we discuss possible ways to instantiate the message mapping H and the underlying non-interactive zero-knowledge (NIZK) proofs performed by the server and the client.

### C.1 Proof by the Server

The NIZK proof, $\pi_s$, (see Figure 10) conducted by the server is the most typical proof needed in lattice-based cryptography, and there are various proof systems that can be used to instantiate it. Some notable ones are the LANES [ALS20, ENS20, LNS20], LANES$^+$ [ESLR23], LNP22 [LNP22] and LaBRADOR [BS23] proof systems. As discussed in Appendix F, we use a lattice-based commitment scheme, and thus, these proof systems can natively support its well-formedness proof. Given we require a relatively large modulus $q$, our experiments show that LaBRADOR in combination with LNP22 to achieve zero-knowledge offers the best communication performance among these options. Thanks to its succinctness features, it also scales very efficiently in the batched setting. Note that even though LaBRADOR as described in [BS23] does not directly achieve zero-knowledge, it is easy to achieve it as discussed in "Zero-Knowledge Property" part of [BS23, Page 3]. In particular, we assume LNP22 is used as the shim protocol to achieve zero-knowledge. We report on the performance results in Section 5.

### C.2 Instantiating the Message Mapping H

In this section, we primarily focus on the *secret-dependent* part of the message mapping H and discuss our main option on how the message mapping $\mathsf{H}\colon x \mapsto \mathbf{B}_x \in R_q^{h\times m}$ can be instantiated. That is, we assume the public input part $t$ to be empty. When the tag $t$ is present, the mapping would simply be computed on $\hat{x} := x\|t$ as $\mathsf{H}(\hat{x})$ and it is easy to verify the computation w.r.t. this input extended with *public* information (which we discuss more in Appendix C.3). There are two main considerations: (i) security of MLWE-PRF-RU w.r.t. to H chosen, and (ii) efficiency of our OPRF, particularly the underlying NIZK proofs especially by the client. The latter is the primary consideration as we believe MLWE-PRF-RU remains secure for all instantiations discussed in this work given the reductions in Section 3.

In line with the MLWE-PRF-RU security reductions in Section 3, we believe that a natural way to instantiate H is to use the mappings in existing lattice-based PRFs so that the SampPRF oracle in MLWE-PRF really serves as an oracle outputting (randomized) PRF samples (in the case of $b = 1$). Therefore, we discuss options based on existing lattice-based PRFs.

**Using the mapping from BP14 PRF [BP14] for $\mathsf{H}$.** The option we see as the most suitable one is using the message mapping employed in BP14 PRF [BP14]. Let $\mathfrak{l} := \lceil \log q \rceil$. Here, the PRF computation involves a gadget matrix $\mathbf{G}$ and its (non-linear) inverse computation $\mathbf{G}^{-1} \colon R_q^{m \times m\mathfrak{l}} \to R_q^{m\mathfrak{l} \times m\mathfrak{l}}$. The $\mathbf{G}^{-1}$ mapping is the standard bit decomposition operation. In this case, we have two matrices $\mathbf{A}_0, \mathbf{A}_1 \xleftarrow{\$} R_q^{m \times m\mathfrak{l}}$ published as public parameters. Then, for $\mathbf{k} \leftarrow \chi_k^m$ for some distribution $\chi_k$ [12] on $R_q$, we compute[13]

$$F_{\mathbf{k}}^{\mathsf{bp}}(x) = \left\lfloor \mathbf{B}_x^{\mathsf{bp}} \cdot \mathbf{k} \right\rceil_p, \tag{28}$$

where the message mapping $\mathsf{H}^{\mathsf{bp}}$ is defined as

$$\mathsf{H}^{\mathsf{bp}} \colon x \mapsto (\mathbf{B}_x^{\mathsf{bp}})^\top := \mathbf{A}_{x_0} \cdot \mathbf{G}^{-1}(\mathbf{A}_{x_1} \cdots \mathbf{G}^{-1}(\mathbf{A}_{x_{L-2}} \cdot \mathbf{G}^{-1}(\mathbf{A}_{x_{L-1}}))). \tag{29}$$

If there is a tag $t$ present, we treat it as the last bits of the input to $\mathsf{H}^{\mathsf{bp}}$. For better efficiency of the OPRF, we can truncate $\mathbf{B}_x^{\mathsf{bp}}$ and just use its first row $\mathbf{B}_x^{\mathsf{bp}}[0]$ for the final PRF computation (as also done in [AG24]). Note that this does not impact security because pseudorandomness of $\lfloor \mathbf{B}_x \cdot \mathbf{k} \rceil_p$ implies pseudorandomness of $\lfloor \mathbf{B}_x[0] \cdot \mathbf{k} \rceil_p$ in general as the latter is just the first coordinate of the former vector. Overall, for the BP14 option, we take the message mapping used in $\mathsf{LeOPaRd}$ as $\mathsf{H} \colon x \mapsto \mathbf{B}_x := \mathbf{B}_x^{\mathsf{bp}}[0] \in R_q^{1 \times m}$, resulting in $h = 1$. As discussed in [BP14], another well-known lattice-based PRF scheme, BMLR13 PRF, [BLMR13] can be seen as a special case of BP14 PRF.

For the client's NIZK proof in this case (discussed further in Appendix C.3), we will need to treat each $\mathbf{B}_i := \mathbf{G}^{-1}(\mathbf{A}_{x_i} \cdot \mathbf{B}_{i+1}) \in R_q^{m\mathfrak{l} \times m\mathfrak{l}}$ for $i = L-1, \ldots, 0$ (with $\mathbf{B}_L := \mathbf{I}_m$) as a variable. Therefore, there are $m^2\mathfrak{l}^2 L$ variable polynomials.

We may also consider a generalized version of this PRF proposal where the gadget matrix works with base $\gamma \geq 2$ (instead of $\gamma = 2$ as before), i.e., $\mathbf{g} = (1, \gamma, \gamma^2, \ldots)$, and $\mathbf{G} = \mathbf{g} \otimes \mathbf{I}$. In this case, we would have that $\mathfrak{l} = \lceil \log_\gamma q \rceil$.

**Using the mapping from BLMR13 PRF [BLMR13] for $\mathsf{H}$.** An other option is the BLMR13 PRF [BLMR13] (particularly its variant over module lattices as described in [SSS23]). In this setting, we have two square matrices $\mathbf{A}_0, \mathbf{A}_1 \in R_q^{m \times m}$ (with binary entries) published as public parameters. To compute the PRF for an input $x = (x_0, \ldots, x_{L-1})$ in binary and $\mathbf{k} \xleftarrow{\$} R_q^m$, we compute

$$F_{\mathbf{k}}^{\mathsf{blmr}}(x) = \left\lfloor \prod_{i=0}^{L-1} \mathbf{A}_{x_i} \cdot \mathbf{k} \right\rceil_p. \tag{30}$$

As a result, the message mapping $\mathsf{H}^{\mathsf{blmr}}$ is defined as

$$\mathsf{H}^{\mathsf{blmr}} : x \mapsto \mathbf{B}_x^{\mathsf{blmr}} = \prod_{i=0}^{L-1} \mathbf{A}_{x_i}. \tag{31}$$

If there is a tag $t$ present, we treat $t$ as the first part of the message in this case. That is, if $t = (t_0, \ldots)$ and $x = (x_0, \ldots)$, then $\mathbf{B}_x^{\mathsf{blmr}} = \prod_i \mathbf{A}_{t_i} \prod_i \mathbf{A}_{x_i}$. As in Appendix C.2, we can truncate the final PRF output and take $\mathbf{B}_x = \mathbf{B}_x^{\mathsf{blmr}}[0] \in R_q^{1 \times m}$ as the first row of $\mathbf{B}_x^{\mathsf{blmr}}$ in $\mathsf{LeOPaRd}$ (i.e., $h = 1$).

For the client's NIZK proof $\pi_c$ (discussed in Appendix C.3), we will need to treat each $\mathbf{A}_{x_i}$ as a set of different variables. It is easy to observe that there are $m^2 L$ variable polynomials in that case. As discussed in [SSS23], we require $m \geq 2$ as otherwise this module variant of the PRF scheme is insecure. In fact, $m = n\lceil \log q \rceil$ is required in the provable security result in [BLMR13] to base pseudorandomness on $\mathsf{MLWE}$ hardness with rank $n$.

[12] Note that in this case we may have small secret key coefficients.

[13] We note that our presentation here is for multiplying the PRF key from the *right* unlike [BP14] where it is from the *left*.

**Using a (symmetric) PRG for $\mathsf{H}$.** Another option one may consider is to use a (symmetric) PRG to instantiate $\mathsf{H}$. In this case, we may simply set $h = 1$ to minimize a dimension of certain matrices/vectors involved in communication and NIZK proofs for improved performance. In this case, the $\mathsf{SampPRF}$ oracle in $\mathsf{MLWE\text{-}PRF}$ would output samples (computationally) indistinguishable from $\mathsf{MLWE}$ samples (based on the PRG pseudorandomness property). One may then employ a suitable zk-SNARK system to instantiate the client's proof $\pi_c$. We do not see this option as competitive (in terms of efficiency) as the prior ones where fully lattice-based tools can be employed for improved compatibility and performance. Therefore, we do not discuss this option in further detail. However, as further progress is made in the development of more efficient zk-SNARK solutions, this option (just like the others) may become more efficient over time.

**Modelling $\mathsf{H}$ as a random oracle.** We also note the option of modeling $\mathsf{H}$ as a random oracle. In this case, we may again simply set $h = 1$ to minimize a dimension of certain matrices/vectors involved in communication and NIZK proofs for improved performance. In this case, the $\mathsf{SampPRF}$ oracle in $\mathsf{MLWE\text{-}PRF}$ would simply output $\mathsf{MLWE}$ samples, and the hardness of $\mathsf{MLWE\text{-}PRF\text{-}RU}$ problem is more easily established based on $\mathsf{MLWE}$ as discussed in Section 3. Assuming $\mathsf{H}$ to be a random oracle would also reduce the control of an OPRF pseudorandomness adversary on how $\mathbf{C}_x$ is computed.

There are two (related) caveats with this choice. Firstly, we would need a NIZK proof that proves pre-image knowledge of a hash function (modelled as a random oracle). There are generic zk-SNARK systems like [COS20] that can achieve this for standard hash function such as SHA3, but they are currently not as efficient as we would like them to be. Alternatively, we may use a more algebraic hash function like Poseidon [GKR+21] to help reduce the cost of the NIZK well-formedness proof. Overall, the first caveat is that the associated message-mapping well-formedness proof is not going to be as efficient.

The second caveat is that we would assume $\mathsf{H}$ to be a random oracle, but then, as we need to prove message-mapping well-formedness via a NIZK proof, we would also need to assume that $\mathsf{H}$ has a compact circuit representation. This is the same issue that has already been encountered in earlier works such as the recursive proof in [COS20], the blind signature in [BLNS23] and the aggregate signature in [ZSE+24]. Similar to earlier works, we can (heuristically) assume that the construction remains secure when the random oracle is instantiated with a hash function and then instantiate the NIZK accordingly.

Overall, while it does not currently appear as a very competitive option in terms of (NIZK) efficiency, we believe that the main advantage of modelling $\mathsf{H}$ as a random oracle arises in helping with the security reductions (as in Section 3) and establishing a tighter (theoretical) bound on an adversary's winning chance in OPRF pseudorandomness.

### C.3 Proof by the Client

In this section, we discuss how the client's NIZK proof can be instantiated. One can observe from the NIZK relation proven by the client (given in Figure 10) that the majority of the proof components are the most common relations proven in lattice-based cryptography (when the commitment scheme is instantiated using a suitable lattice-based extractable commitment scheme). There is perhaps one exception to this: (i) proving $\mathbf{B}_{x,t} \coloneqq \mathsf{H}(x,t)$. Accordingly, we next discuss the proof of $\mathbf{B}_{x,t} \coloneqq \mathsf{H}(x,t)$. Depending on the instantiation of the message mapping $\mathsf{H}$, the client's proof may vary significantly. We look more closely at the instantiation of $\mathsf{H}$ discussed in Appendix C.2.

**NIZK for BP14 PRF mapping.** Recall we have $h = 1$ and $\mathfrak{l} \coloneqq \lceil \log_\gamma q \rceil$ for some $\gamma \geq 2$. We first discuss the case when no tag $t$ is present, i.e., $\mathbf{B}_x \coloneqq \mathsf{H}(x)$. The case of $\mathbf{B}_{x,t} \coloneqq \mathsf{H}(x,t)$ is discussed at the end of this section. For a known $\mathbf{A}_r$ and $\mathbf{C}_x$, the prover wants to prove knowledge of short matrix $\mathbf{R}$ and a bit-string $x = (x_0, x_1, \ldots, x_{L-1}) \in \{0,1\}^L$ such that $\mathbf{C}_x = \mathbf{R}\mathbf{A}_r + \mathbf{B}_x \bmod q$ where $\mathbf{B}_x \coloneqq \mathbf{B}_x^{\mathsf{bp}}[0]$ and $\mathbf{B}_x^{\mathsf{bp}}$ is defined in (29).

Using a similar strategy as in [ADDS21] (see "**Proof system 1: Proofs of Masked Partial PRF Computation**"), define variables $\mathbf{B}_i \in R_q^{m\mathfrak{l} \times m\mathfrak{l}}$ for $i = L-1, \ldots, 0$ as $\mathbf{B}_{L-1} \coloneqq \mathbf{G}^{-1}(\mathbf{A}_{x_{L-1}})$ and $\mathbf{B}_i \coloneqq \mathbf{G}^{-1}(\mathbf{A}_{x_i} \cdot \mathbf{B}_{i+1})$ for $i = L-2, \ldots, 0$. Using this, we have $\mathbf{B}_x = (\mathbf{G} \cdot \mathbf{B}_0)^\top[0]$. Then the statement being

proven now can be transformed to prove the following system with $\mathbf{B}_L \coloneqq \mathbf{I}$:

$$\begin{aligned}\mathbf{G} \cdot \mathbf{B}_i &= \mathbf{A}_{x_i} \cdot \mathbf{B}_{i+1}, \quad i = 0, \ldots, L-1\\ \mathbf{C}_x &= \mathbf{R}\mathbf{A}_r + (\mathbf{G} \cdot \mathbf{B}_0)^\top[0].\end{aligned}$$

This system of equations above is not linear due to the $x_i$ and $\mathbf{B}_{i+1}$ terms and the fact that $\mathbf{G}^{-1}$ is not a linear operator. We can manipulate the system of equations above to obtain

$$\begin{aligned}\mathbf{G} \cdot \mathbf{B}_{L-1} &= \mathbf{A}_0 \cdot (1 - x_{L-1}) + \mathbf{A}_1 \cdot x_{L-1},\\ \mathbf{G} \cdot \mathbf{B}_{L-2} &= \mathbf{A}_0 \cdot (1 - x_{L-2}) \cdot \mathbf{B}_{L-1} + \mathbf{A}_1 \cdot x_{L-2} \cdot \mathbf{B}_{L-1},\\ &\vdots\\ \mathbf{G} \cdot \mathbf{B}_0 &= \mathbf{A}_0 \cdot (1 - x_0) \cdot \mathbf{B}_1 + \mathbf{A}_1 \cdot x_0 \cdot \mathbf{B}_1,\\ \mathbf{C}_x &= \mathbf{R}\mathbf{A}_r + (\mathbf{G} \cdot \mathbf{B}_0)^\top[0].\end{aligned}$$

The required proof now boils down to proving that a sequence of quadratic equations holds (since $\mathbf{A}_i$'s are public), which can be easily handled by LaBRADOR [BS23]. Observe that the witness dimension (over $R_q$) is in the order of $m^2\mathfrak{l}^2 L$. In our parameter settings (see Section 5), the largest $m$ is around 64 and the largest $\log_2 q$ is less than 128; hence $m^2\mathfrak{l}^2 L = m^2 L(\log_2 q/\log_2 \gamma)^2 \leq 2^{32}$ for $L = 64$. By choosing larger $\gamma$ or smaller rank parameter $m$, this witness dimension can be significantly reduced. In [NS24], the authors report a 53 KB LaBRADOR-based proof size for a polynomial evaluation proof of degree $2^{30}$ with all running times around a couple of minutes or less. Therefore, we believe the proof we require will be reasonably efficient.

Note that when a tag $t$ is used in computing $\mathbf{B}_{x,t} \coloneqq \mathsf{H}(x, t)$ (i.e., $\mathbf{B}_{\hat{x}} \coloneqq \mathsf{H}(\hat{x})$ for $\hat{x} \coloneqq x\|t$), the above system of equations will have a minor change in the first expression such that we will have $\mathbf{G} \cdot \mathbf{B}_{L-1} = \mathbf{A}_0 \cdot (1 - x_{L-1}) \cdot \mathbf{B}_t + \mathbf{A}_1 \cdot x_{L-1} \cdot \mathbf{B}_t$ for some *public* matrix $\mathbf{B}_t$ dependent on the tag $t$. The NIZK proof can equivalently prove this similar system of equations.

**NIZK for BLMR13 PRF mapping.** Here, for a known $\mathbf{A}_r$ and $\mathbf{C}_x$, the prover (client) wants to prove knowledge of short $\mathbf{R}$ and a bit-string $x = (x_0, x_1, \ldots, x_{L-1}) \in \{0,1\}^L$ such that $\mathbf{C}_x = \mathbf{R}\mathbf{A}_r + \mathbf{B}_x \bmod q$ where $\mathbf{B}_x \coloneqq \mathbf{B}_x^{\mathsf{blmr}}[0]$ and $\mathbf{B}_x^{\mathsf{blmr}}$ is defined in (31). To be more specific, $\mathbf{B}_x^{\mathsf{blmr}} = \prod_{i=0}^{L-1} \mathbf{A}_{x_i}$, where $\mathbf{A}_0$, $\mathbf{A}_1$ are two public matrices. Now, define variables $\mathbf{B}_i \in R_q^{m \times m}$ for $i = L-1, \ldots, 0$ as $\mathbf{B}_i \coloneqq \prod_{j=0}^{i} \mathbf{A}_{x_i}$. Then, the statement being proved can be expressed as follows.

$$\begin{aligned}\mathbf{B}_0 &= \mathbf{A}_{x_0},\\ \mathbf{B}_{i+1} &= \mathbf{B}_i \cdot \mathbf{A}_{x_{i+1}}, \quad i = 0, \ldots, L-2,\\ \mathbf{C}_x &= \mathbf{R}\mathbf{A}_r + \mathbf{B}_{L-1}.\end{aligned}$$

This system of equations above is not linear due to the $x_i$ and $\mathbf{B}_{i+1}$ terms. We can represent the terms $\mathbf{B}_i \cdot \mathbf{A}_{x_{i+1}}$ as $\mathbf{B}_i \cdot \mathbf{A}_0 \cdot (1 - x_{i+1}) + \mathbf{B}_i \cdot \mathbf{A}_1 \cdot x_{i+1}$, then we obtain

$$\begin{aligned}\mathbf{B}_0 &= \mathbf{A}_0 \cdot (1 - x_0) + \mathbf{A}_1 \cdot x_0,\\ \mathbf{B}_1 &= \mathbf{B}_0 \cdot \mathbf{A}_0 \cdot (1 - x_1) + \mathbf{B}_0 \cdot \mathbf{A}_1 \cdot x_1,\\ &\vdots\\ \mathbf{B}_{L-1} &= \mathbf{B}_{L-2} \cdot \mathbf{A}_0 \cdot (1 - x_{L-1}) + \mathbf{B}_{L-2} \cdot \mathbf{A}_1 \cdot x_{L-1},\\ \mathbf{C}_x &= \mathbf{R}\mathbf{A}_r + \mathbf{B}_{L-1}.\end{aligned}$$

Now the required proof boils down to proving that a sequence of quadratic equations holds, which can again be easily handled by LaBRADOR [BS23]. Observe that the witness dimension (over $R_q$) is again in the order of $m^2 L$. Note also that when a tag $t$ is present, we will have a similar situation as described before for the BP14 PRF, where some *public* matrix $\mathbf{B}_t$ will appear in the first expression describing $\mathbf{B}_0$.

# D Deferred Definitions and Proofs

In the security proofs, we use various tables (denoted such as $\mathcal{T}$) to store values. For ease of presentation, we write $\mathcal{T}[x] = \perp$ to mean that a value, $x$, is not stored/defined in the table $\mathcal{T}$ before. For a newly-initialized table, $\mathcal{T}[x] = \perp$ for all $x$.

## D.1 Proof of Hardness Reduction MLWE → MLWE-PRF for BP14 $\mathsf{H}^{\mathsf{bp}}$

**Theorem 4** (MLWE → MLWE-PRF **for BP14** H)**.**

*Let* $\mathsf{H}^{\mathsf{bp}}$ *be the BP14 function family with $L$-bit input and error distribution* $\chi_{\mathbf{A_0},\mathbf{A_1},x,s_0}$ *as in Def. 13. Let* $\chi_k := \mathcal{U}(R_q)$, $\chi_0 := \mathcal{D}_{\mathbb{Z}^d,s_0}$ *and* $\chi_{0,B} := \chi_{\mathbf{A_0},\mathbf{A_1},x,s_0}$. *The* $\mathsf{MLWE\text{-}PRF}_{\mathsf{H}^{\mathsf{bp}},Q_\mathsf{P},Q_\mathsf{M},q,m,h,L,\chi_0,\chi_{0,B},\chi_k}$ *assumption holds if the* $\mathsf{MLWE}_{2m\mathfrak{l}+Q_\mathsf{M},q,m,1,\chi_0,\chi_k}$ *and* $\mathsf{MLWE}_{2m\mathfrak{l},q,m,Q_\mathsf{P},\chi_0,\chi_k}$ *assumptions hold.*

*More precisely, for any* PPT *adversary* $\mathcal{A}$ *against the* $\mathsf{MLWE\text{-}PRF}_{\mathsf{H}^{\mathsf{bp}},Q_\mathsf{P},Q_\mathsf{M},q,m,h,L,\chi_0,\chi_{0,B},\chi_k}$ *problem, there exist* PPT *adversaries* $\mathcal{B},\overline{\mathcal{B}}$ *against the* $\mathsf{MLWE}_{2m\mathfrak{l}+Q_\mathsf{M},q,m,1,\chi_0,\chi_k}$ *and* $\mathsf{MLWE}_{2m\mathfrak{l},q,m,Q_\mathsf{P},\chi_0,\chi_k}$ *problems respectively, such that*

$$\mathsf{Adv}^{\mathsf{MLWE\text{-}PRF}}_{\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{MLWE}}_{\mathcal{B}}(\lambda) + (L-1)\cdot \mathsf{Adv}^{\mathsf{MLWE}}_{\overline{\mathcal{B}}}(\lambda)\,.$$

*Proof (of Theorem 4).*

Let $\mathcal{A}$ denote an adversary against MLWE-PRF. As outlined in the main text, the security proof follows in a series of hybrid games similar to the reduction in [BP14], where we start with the hybrid $\mathbf{Hybrid_{0,1}}$ in which the view of $\mathcal{A}$ is the same as in the pseudorandom ($b = 1$) experiment $\mathsf{MLWE\text{-}PRF}^1$ defined in Figure 4 with $\mathsf{H} = \mathsf{H}^{\mathsf{bp}}$, and we end with the hybrid $\mathbf{Hybrid_{L-1,0}}$, in which the view of $\mathcal{A}$ is the same as in the random ($b = 0$) experiment $\mathsf{MLWE\text{-}PRF}^0$. For $i \in \{0,\ldots,L-1\}$ and $\eta \in \{0,1\}$, We say that $\mathbf{Hybrid_{i,\eta}}$ returns 1 if $\mathcal{A}$ outputs $b' = 1$.

$\mathbf{Hybrid_{0,1}}$: This corresponds to the $\mathsf{MLWE\text{-}PRF}^1$ game (Definition 11), and the interaction follows as in Figure 11.

$\mathsf{MLWE\text{-}PRF}^b_{\mathsf{H},Q_\mathsf{P},Q_\mathsf{M},q,m,m\mathfrak{l},L,\overrightarrow{\chi}}(\lambda)$

$\mathbf{k} \xleftarrow{\$} \chi_k^m$
$\mathcal{Y} := \emptyset$ // Initialize empty table
$\mathcal{E} := \emptyset$ // Initialize empty table
$b' \leftarrow \mathcal{A}^{\mathsf{SampPRF},\mathsf{SampMLWE}}(1^\lambda)$
**return** $b = b'$

$\mathsf{SampMLWE}()$

$\mathbf{a} \xleftarrow{\$} R_q^m, e \xleftarrow{\$} \chi_0$
$c := \mathbf{k}^\top \mathbf{a} + e \in R_q$
**return** $(\mathbf{a}, c)$

$\mathsf{SampPRF}(x)$

**if** $\mathcal{Y}[x] = \perp$ **then**
  $(\mathbf{B}_x)^\top := (\mathsf{H}^{\mathsf{bp}}_{\mathbf{A}_0,\mathbf{A}_1}(x))^\top \in R_q^{m\times m\mathfrak{l}}$
  // $= \mathbf{A}_{x_0}\cdot\mathbf{G}^{-1}(\mathbf{A}_{x_1}\cdots\mathbf{G}^{-1}(\mathbf{A}_{x_{L-1}})))$
  // $= \mathbf{A}_{x_0}\cdot\mathbf{B}_{x/1}$
  $\forall j \in \{0,\ldots,L-1\}$:
  **if** $\mathcal{E}[x\setminus(j+1)] = \perp$ **then**
    $\mathcal{E}[x\setminus(j+1)] \xleftarrow{\$} \chi_0^{m\mathfrak{l}}$
  $\mathbf{e}_{x,j} := \mathcal{E}[x\setminus(j+1)]$
$\mathbf{e}_x^\top := \sum_{j=0}^{L-2} \mathbf{e}_{x,j}^\top\cdot\mathbf{B}_{x/(j+1)} + \mathbf{e}_{x,L-1}^\top$
  // $= \mathbf{e}_{x_0}^\top\cdot\mathbf{B}_{x/1} + \mathbf{e}_{x/1}^\top$
  $\mathcal{Y}[x]^\top := \mathbf{k}^\top\mathbf{B}_x + \mathbf{e}_x^\top \in R_q^{m\mathfrak{l}}$
  // $= (\mathbf{k}^\top\mathbf{A}_{x_0} + \mathbf{e}_{x,0}^\top)\cdot\mathbf{B}_{x/1} + \mathbf{e}_{x/1}^\top$
**return** $\mathcal{Y}[x]$

Fig. 11: $\mathbf{Hybrid_{0,1}}$ in which $\mathcal{A}$'s view is as in experiment $\mathsf{MLWE\text{-}PRF}^1$

Since the view of $\mathcal{A}$ in $\mathbf{Hybrid_{0,1}}$ is the same as in the pseudorandom ($b = 1$) experiment $\mathsf{MLWE\text{-}PRF}^1$ defined in Figure 4 with $\mathsf{H} = \mathsf{H}^{\mathsf{bp}}$, it follows that

$$\Pr[\mathbf{Hybrid_{0,1}} = 1] = \Pr\left[\mathsf{MLWE\text{-}PRF}^1(\lambda) = 1\right].$$

**$\mathbf{Hybrid_{0,0}}$**: In this hybrid, we make the following two changes:

- In $\mathsf{SampPRF}$, we replace $(\mathbf{k}^\top \mathbf{A}_{x_0} + \mathbf{e}_{x,0}^\top)$ in the computation of $\mathcal{Y}[x]$ with a uniformly random $\mathbf{u}_{x_0}^\top$ in $R_q^{m\mathfrak{l}}$, where the pair $\mathbf{u}_{x_0}^\top$ for $x_0 \in \{0,1\}$ are independently sampled.
- In $\mathsf{SampMLWE}$, we replace $\mathbf{k}^\top \mathbf{a} + e$ in the computation of $c$ with a uniform and independent ring element $c \xleftarrow{\$} R_q$.

This hybrid is shown in Figure 12.

$\mathsf{MLWE\text{-}PRF}^b_{\mathsf{H},Q_\mathsf{P},Q_\mathsf{M},q,m,m\mathfrak{l},L,\vec{\chi}}(\lambda)$

1 : $\mathbf{k} \xleftarrow{\$} \chi_k^m$
2 : $\mathcal{Y} := \emptyset$ // Initialize empty table
3 : $\mathcal{E} := \emptyset$ // Initialize empty table
4 : $\mathcal{U} := \emptyset$ // Initialize empty table
5 : $b' \leftarrow \mathcal{A}^{\mathsf{SampPRF},\mathsf{SampMLWE}}(1^\lambda)$
6 : **return** $b = b'$

$\mathsf{SampMLWE}()$

1 : $\mathbf{a} \xleftarrow{\$} R_q^m, e \xleftarrow{\$} \chi_0$
2 : $c \xleftarrow{\$} R_q$
3 : **return** $(\mathbf{a}, c)$

$\mathsf{SampPRF}(x)$

1 : **if** $\mathcal{Y}[x] = \bot$ **then**
2 : $\quad (\mathbf{B}_x)^\top := (\mathsf{H}^{\mathsf{bp}}_{\mathbf{A}_0,\mathbf{A}_1}(x))^\top \in R_q^{m \times m\mathfrak{l}}$
3 : $\quad$ // $= \mathbf{A}_{x_0} \cdot \mathbf{G}^{-1}(\mathbf{A}_{x_1} \cdots \mathbf{G}^{-1}(\mathbf{A}_{x_{L-1}}))$
4 : $\quad$ // $= \mathbf{A}_{x_0} \cdot \mathbf{B}_{x/1}$
5 : $\forall j \in \{0, \ldots, L-1\}$:
6 : $\quad$ **if** $\mathcal{E}[x \setminus (j+1)] = \bot$ **then**
7 : $\quad\quad \mathcal{E}[x \setminus (j+1)] \xleftarrow{\$} \chi_0^{m\mathfrak{l}}$
8 : $\quad \mathbf{e}_{x,j} := \mathcal{E}[x \setminus (j+1)]$
9 : $\mathbf{e}_x^\top := \sum_{j=0}^{L-2} \mathbf{e}_{x,j}^\top \cdot \mathbf{B}_{x/(j+1)} + \mathbf{e}_{x,L-1}^\top$
10 : $\quad$ // $= \mathbf{e}_{x_0}^\top \cdot \mathbf{B}_{x/1} + \mathbf{e}_{x/1}^\top$
11 : $\quad$ **if** $\mathcal{U}[x_0] = \bot$ **then**
12 : $\quad\quad \mathcal{U}[x_0] \xleftarrow{\$} R_q^{m\mathfrak{l}}$
13 : $\quad \mathbf{u}_{x_0} := \mathcal{U}[x_0]$
14 : $\quad \mathcal{Y}[x]^\top := \mathbf{u}_{x_0}^\top \cdot \mathbf{B}_{x/1} + \mathbf{e}_{x/1}^\top \in R_q^{m\mathfrak{l}}$
15 : $\quad$ // $= \mathbf{A}_{x_0} \cdot \mathbf{G}^{-1}(\mathbf{A}_{x_1} \cdots \mathbf{G}^{-1}(\mathbf{A}_{x_{L-1}})))$.
16 : **return** $\mathcal{Y}[x]$

Fig. 12: $\mathbf{Hybrid_{0,0}}$ of the $\mathsf{MLWE}$ to $\mathsf{MLWE\text{-}PRF}$ reduction proof

We now construct an adversary $\mathcal{B}$ against the $\mathsf{MLWE}_{2m\mathfrak{l}+Q_\mathsf{M},q,m,1,\chi_0,\chi_k}$ problem, such that

$$|\Pr[\mathbf{Hybrid_{0,0}} = 1] - \Pr[\mathbf{Hybrid_{0,1}} = 1]| = \mathsf{Adv}^{\mathsf{MLWE}}_{\mathcal{B}}(\lambda). \tag{32}$$

On input MLWE instance $(\mathbf{A}, \mathbf{b}) \in R_q^{(2m\mathfrak{l}+Q_\mathsf{M}) \times m} \times R_q^{2m\mathfrak{l}+Q_\mathsf{M}}$, adversary $\mathcal{B}$ parses them as $\mathbf{A}^T := (\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_\mathsf{M}) \in R_q^{m \times m\mathfrak{l}} \times R_q^{m \times m\mathfrak{l}} \times R_q^{m \times Q_\mathsf{M}}$ and $\mathbf{b}^T := (\mathbf{b}_0^\top, \mathbf{b}_1^\top, \mathbf{b}_\mathsf{M}^\top) \in R_q^{m\mathfrak{l}} \times R_q^{m\mathfrak{l}} \times R_q^{Q_\mathsf{M}}$, and runs $\mathcal{A}$ simulating its oracles as in $\mathbf{Hybrid_{0,0}}$ but with the following modifications:

- In $\mathsf{SampPRF}$, we replace assignment $\mathcal{U}[x_0] \xleftarrow{\$} R_q^{m\mathfrak{l}}$ in line 12 with the assignment $\mathcal{U}[x_0] := \mathbf{b}_{x_0} \in R_q^{m\mathfrak{l}}$.
- In $\mathsf{SampMLWE}$, for the $i$'th oracle call of $\mathcal{A}$ (with $i \in [Q_\mathsf{M}]$), we replace assignment $c \xleftarrow{\$} R_q$ in line 2 with the assignment $c := b_{\mathsf{M},i}$, where $b_{\mathsf{M},i} \in R_q$ denotes the $i$th $R_q$-coordinate of $\mathbf{b}_\mathsf{M}^\top \in R_q^{Q_\mathsf{M}}$.

When $\mathcal{A}$ terminates and outputs $b'$, $\mathcal{B}$ terminates and outputs $b'$.

Observe that if the input MLWE instance comes from the 'Real' MLWE distribution ($b = 1$) with secret vector $\mathbf{s}$ sampled from $\chi_k^m$ and error vector $\mathbf{e}^\top = (\mathbf{e}_0^\top, \mathbf{e}_1^\top, \mathbf{e}_{\mathsf{M}}^\top) \in R_q^{m\mathfrak{l}} \times R_q^{m\mathfrak{l}} \times R_q^{Q_{\mathsf{M}}}$ sampled from $\chi_e^{2m\mathfrak{l}+Q_{\mathsf{M}}}$, then:

– In $\mathsf{SampPRF}$ queries,

$$\mathbf{u}_{x_0}^T = \mathbf{b}_{x_0}^\top = (\mathbf{s}^\top \mathbf{A}_{x_0} + \mathbf{e}_{x,0}^\top),$$

so

$$\mathcal{Y}[x]^\top = (\mathbf{s}^\top \mathbf{A}_{x_0} + \mathbf{e}_{x,0}^\top) \cdot \mathbf{B}_{x/1} + \mathbf{e}_{x/1}^\top$$

– In the $i$'th $\mathsf{SampMLWE}$ query,

$$c := b_{\mathsf{M},i} = \mathbf{s}^\top \mathbf{a}_{\mathsf{M},i} + e_{\mathsf{M},i},$$

where $\mathbf{a}_{\mathsf{M},i}$ and $e_{\mathsf{M},i}$ denote the $i$th column and coordinate of $\mathbf{A}_{\mathsf{M}}$ and $\mathbf{e}_{\mathsf{M}}^\top$, respectively. Therefore, in this case, $\mathcal{B}$ simulates to $\mathcal{A}$ the same view as in $\mathbf{Hybrid_{0,1}}$, with $\mathbf{k} := \mathbf{s}$.

On the other hand, in the case that the input MLWE instance comes from the 'Random' MLWE distribution ($b = 0$), then $\mathbf{u}_{x_0}^T = \mathbf{b}_{x_0}^\top$ are uniform in $R_q^{m\mathfrak{l}}$ and independent for $x_0 \in \{0, 1\}$ and $b_{\mathsf{M},i}$ are also uniform in $R_q$ and independent. Therefore, in this case, $\mathcal{B}$ simulates to $\mathcal{A}$ the same view as in $\mathbf{Hybrid_{0,0}}$.

The above two observations establish the claim (32).

Starting with $i = 1$, we now continue inductively to similarly define $(L-1)$ pairs of hybrids $(\mathbf{Hybrid_{i,1}}, \mathbf{Hybrid_{i,0}})$ for $i = 1, \ldots, L-1$ as follows.

$\mathbf{Hybrid_{i,1}}$: Compared to the previous $\mathbf{Hybrid_{i-1,0}}$ (i.e. $\mathbf{Hybrid_{0,0}}$ in the induction base case $i = 1$), we make the following syntactic change:

– In $\mathsf{SampPRF}$, we rewrite $\mathbf{u}_{x\setminus i}^\top$ as $\mathbf{u}_{x\setminus i}^\top := \mathbf{k}_{x\setminus i}^\top \cdot \mathbf{G} + \mathbf{v}_{x\setminus i}^\top$, where $\mathbf{k}_{x\setminus i}^\top$ is sampled uniformly at random from $R_q^m$ (i.e. the distribution $\chi_k^m$) and $\mathbf{v}_{x\setminus i}^\top$ is sampled uniformly at random from $R_q^{m\mathfrak{l}}/(\mathbf{G}R_q^m)$. (recall that $x \setminus i := x_0 x_1 \ldots, x_{i-1}$ denotes the $i$-bit prefix of $x$ and is equal to $x_0$ in the induction base case $i = 1$).

This hybrid is shown in Figure 13.

Assume the induction hypothesis for $i \geq 1$, namely that in the previous game $\mathbf{Hybrid_{i-1,0}}$, the $\mathsf{SampPRF}$ output has the form

$$\mathcal{Y}[x]^\top := \mathbf{u}_{x\setminus i}^\top \cdot \mathbf{B}_{x/i} + \mathbf{e}_{x/i}^\top + \mathbf{v}_{x,i-1}^\top \in R_q^{m\mathfrak{l}}, \tag{33}$$

with $\mathbf{u}_{x\setminus i} := \mathcal{U}[x \setminus i]$ uniform in $R_q^{m\mathfrak{l}}$ and independent for distinct $i$-bit prefixes $x \setminus i$ of $x$ (note that this is satisfied in the induction base case $i = 1$ by the game $\mathbf{Hybrid_{0,0}}$ with $\mathbf{v}_{x,0}^\top := \mathbf{0}$). Then, the change in the game $\mathbf{Hybrid_{i,1}}$ is syntactic only, since the distribution of $\mathbf{u}_{x\setminus i} := \mathbf{k}_{x\setminus i}^\top \cdot \mathbf{G} + \mathbf{v}_{x\setminus i}^\top$ in this game is still uniform uniform in $R_q^{m\mathfrak{l}}$ and independent for distinct $i$-bit prefixes $x \setminus i$ of $x$, due to the uniformly random choice of $\mathbf{k}_{x\setminus i}^\top$ in $R_q^m$ and $\mathbf{v}_{x\setminus i}^\top$ in $R_q^{m\mathfrak{l}}/(\mathbf{G}R_q^m)$. Therefore, the view of $\mathcal{A}$ stays the same as in the previous game, and we have

$$|\Pr[\mathbf{Hybrid_{i,1}} = 1] - \Pr[\mathbf{Hybrid_{i-1,0}} = 1]| = 0. \tag{34}$$

Furthermore, in this game $\mathbf{Hybrid_{i,1}}$, the $\mathsf{SampPRF}$ output has the form

$$\mathcal{Y}[x]^\top := (\mathbf{k}_{x\setminus i}^\top \mathbf{A}_{x_i} + \mathbf{e}_{x,i}^\top) \cdot \mathbf{B}_{x/(i+1)} + \mathbf{e}_{x/(i+1)}^\top + \mathbf{v}_{x,i}^\top, \tag{35}$$

since $\mathbf{B}_{x/i} := \mathbf{G}^{-1}(\mathbf{A}_{x_i} \cdots \mathbf{G}^{-1}(\mathbf{A}_{x_{L-1}}))$ so $\mathbf{k}_{x\setminus i}^\top \mathbf{G} \mathbf{B}_{x/i} = \mathbf{k}_{x\setminus i}^\top \mathbf{A}_{x_i} \mathbf{B}_{x/(i+1)}$, $\mathbf{e}_{x/i}^\top := \mathbf{e}_{x,i}^\top \mathbf{B}_{x/(i+1)} + \mathbf{e}_{x/(i+1)}^\top$ and $\mathbf{v}_{x,i}^\top := \mathbf{v}_{x\setminus i}^\top \mathbf{B}_{x/i} + \mathbf{v}_{x,i-1}^\top$.

$\mathbf{Hybrid_{i,0}}$: In this hybrid, we make the following change:

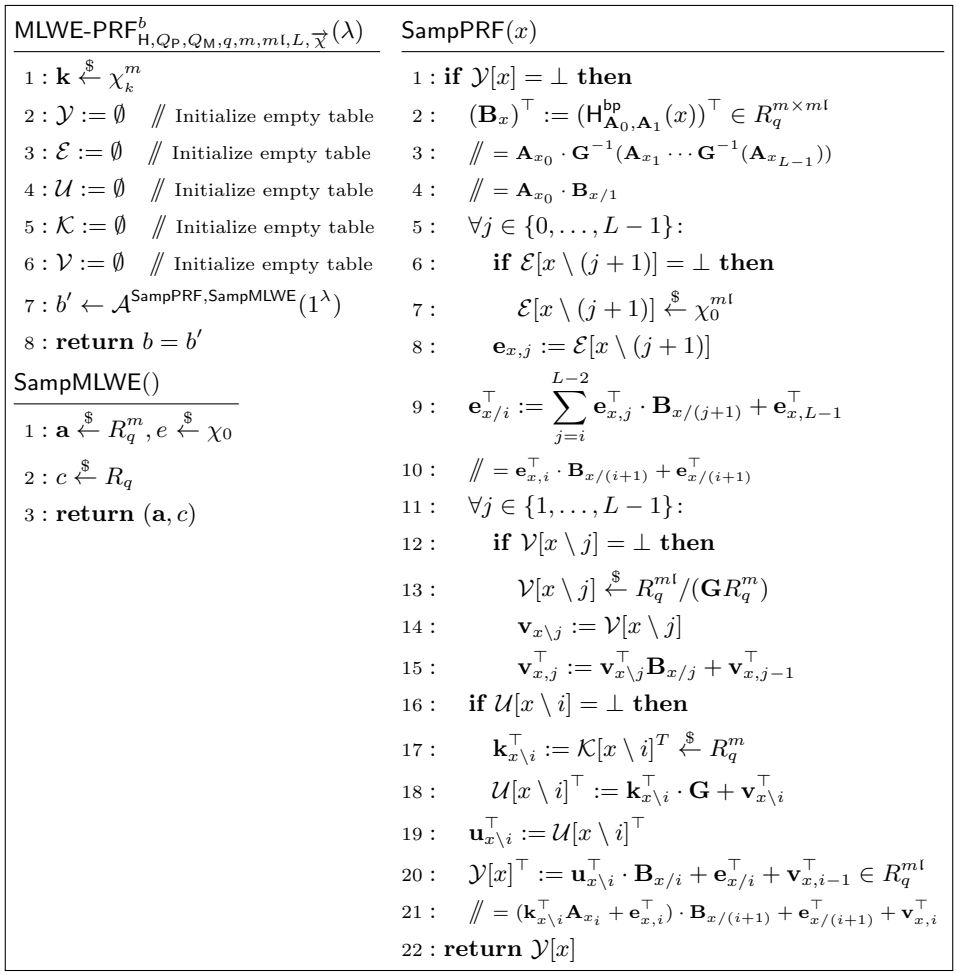


Fig. 13: $\mathbf{Hybrid_{i,1}}$ of the MLWE to MLWE-PRF reduction proof

$\mathsf{MLWE\text{-}PRF}^{b}_{\mathsf{H},Q_\mathsf{P},Q_\mathsf{M},q,m,m\mathfrak{l},L,\vec{\chi}}(\lambda)$

$\mathbf{k} \xleftarrow{\$} \chi_k^m$
$\mathcal{Y} := \emptyset$ // Initialize empty table
$\mathcal{E} := \emptyset$ // Initialize empty table
$\mathcal{U} := \emptyset$ // Initialize empty table
$\mathcal{K} := \emptyset$ // Initialize empty table
$\mathcal{V} := \emptyset$ // Initialize empty table
$b' \leftarrow \mathcal{A}^{\mathsf{SampPRF},\mathsf{SampMLWE}}(1^\lambda)$
**return** $b = b'$

$\mathsf{SampMLWE}()$

$\mathbf{a} \xleftarrow{\$} R_q^m, e \xleftarrow{\$} \chi_0$
$c \xleftarrow{\$} R_q$
**return** $(\mathbf{a}, c)$

$\mathsf{SampPRF}(x)$

**if** $\mathcal{Y}[x] = \perp$ **then**
  $(\mathbf{B}_x)^\top := (\mathsf{H}^{\mathsf{bp}}_{\mathbf{A}_0,\mathbf{A}_1}(x))^\top \in R_q^{m \times m\mathfrak{l}}$
  // $= \mathbf{A}_{x_0} \cdot \mathbf{G}^{-1}(\mathbf{A}_{x_1} \cdots \mathbf{G}^{-1}(\mathbf{A}_{x_{L-1}}))$
  // $= \mathbf{A}_{x_0} \cdot \mathbf{B}_{x/1}$
  $\forall j \in \{0, \ldots, L-1\}$:
    **if** $\mathcal{E}[x \setminus (j+1)] = \perp$ **then**
      $\mathcal{E}[x \setminus (j+1)] \xleftarrow{\$} \chi_0^{m\mathfrak{l}}$
    $\mathbf{e}_{x,j} := \mathcal{E}[x \setminus (j+1)]$
  $\mathbf{e}^\top_{x/i} := \sum_{j=i}^{L-2} \mathbf{e}^\top_{x,j} \cdot \mathbf{B}_{x/(j+1)} + \mathbf{e}^\top_{x,L-1}$
  // $= \mathbf{e}^\top_{x,i} \cdot \mathbf{B}_{x/(i+1)} + \mathbf{e}^\top_{x/(i+1)}$
  $\forall j \in \{1, \ldots, L-1\}$:
    **if** $\mathcal{V}[x \setminus j] = \perp$ **then**
      $\mathcal{V}[x \setminus j] \xleftarrow{\$} R_q^{m\mathfrak{l}}/(\mathbf{G}R_q^m)$
      $\mathbf{v}_{x\setminus j} := \mathcal{V}[x \setminus j]$
      $\mathbf{v}^\top_{x,j} := \mathbf{v}^\top_{x\setminus j}\mathbf{B}_{x/j} + \mathbf{v}^\top_{x,j-1}$
  **if** $\mathcal{U}[x \setminus (i+1)] = \perp$ **then**
    $\mathcal{U}[x \setminus (i+1)] \xleftarrow{\$} R_q^{m\mathfrak{l}}$
  $\mathbf{u}_{x\setminus(i+1)} := \mathcal{U}[x \setminus (i+1)]$
  $\mathcal{Y}[x]^\top := \mathbf{u}^\top_{x\setminus(i+1)} \cdot \mathbf{B}_{x/(i+1)} + \mathbf{e}^\top_{x/(i+1)} + \mathbf{v}^\top_{x,i} \in R_q^{m\mathfrak{l}}$
**return** $\mathcal{Y}[x]$

Fig. 14: $\mathbf{Hybrid_{i,0}}$ of the MLWE to MLWE-PRF reduction proof

– In $\mathsf{SampPRF}$, we replace $(\mathbf{k}_{x\setminus i}^\top \mathbf{A}_{x_i} + \mathbf{e}_{x,i}^\top)$ in the computation of $\mathcal{Y}[x]$ with a uniformly random $\mathbf{u}_{x\setminus(i+1)}^\top$ in $R_q^{m\mathfrak{l}}$, where the $\mathbf{u}_{x\setminus(i+1)}^\top$ is independently sampled for each distinct $i$-bit prefix $x \setminus (i+1) \in \{0,1\}^i$.

This hybrid is shown in Figure 14.

Similarly to $\mathbf{Hybrid_{0,0}}$, we now construct an adversary $\mathcal{B}_i$ against the $\mathsf{MLWE}_{2m\mathfrak{l},q,m,Q_\mathsf{P},\chi_0,\chi_k}$ problem, such that

$$|\Pr[\mathbf{Hybrid_{i,0}} = 1] - \Pr[\mathbf{Hybrid_{i,1}} = 1]| = \mathsf{Adv}^{\mathsf{MLWE}}_{\mathcal{B}_i}(\lambda). \tag{36}$$

On input MLWE instance $(\mathbf{A}, \mathbf{B}) \in R_q^{2m\mathfrak{l}\times m} \times R_q^{2m\mathfrak{l}\times Q_\mathsf{P}}$, adversary $\mathcal{B}_i$ parses them as $\mathbf{A}^T := (\mathbf{A}_0, \mathbf{A}_1) \in R_q^{m\times m\mathfrak{l}} \times R_q^{m\times m\mathfrak{l}}$ and $\mathbf{B}^T := (\mathbf{B}_0^\top, \mathbf{B}_1^\top) \in R_q^{Q_\mathsf{P}\times m\mathfrak{l}} \times R_q^{Q_\mathsf{P}\times m\mathfrak{l}}$, and runs $\mathcal{A}$ simulating its oracles as in $\mathbf{Hybrid_{i,0}}$ but with the following modification:

– In $\mathsf{SampPRF}$, we replace assignment $\mathcal{U}[x \setminus (i+1)] \xleftarrow{\$} R_q^{m\mathfrak{l}}$ in line 17 with the assignment $\mathcal{U}[x \setminus (i+1)] := \mathbf{B}_{x_i}^\top[\mathsf{qind}(x \setminus i)] \in R_q^{m\mathfrak{l}}$, where $\mathbf{B}_{x_i}^\top[j] \in R_q^{m\mathfrak{l}}$ denotes the $j$th row of $\mathbf{B}_{x_i}^\top$ for $j \in [Q_\mathsf{P}]$ and $\mathsf{qind}(x \setminus i)$ returns the index $j \in [Q_\mathsf{P}]$ of the first query $x'$ to $\mathsf{SampPRF}$ which has the same $i$-bit prefix as $x$ (i.e. $x' \setminus i = x \setminus i$).

When $\mathcal{A}$ terminates and outputs $b'$, $\mathcal{B}$ terminates and outputs $b'$.

Observe that if the input MLWE instance comes from the 'Real' (multi-secret) MLWE distribution ($b = 1$), then for $j \in [Q_\mathsf{P}]$, the $j$th row of $\mathbf{B}_{x_i}^\top$ has the form $\mathbf{B}_{x_i}^\top[j] = \mathbf{S}^\top[j]\mathbf{A}_{x_i} + \mathbf{E}_{x_i}^\top[j]$, where the $j$th MLWE secret vector $\mathbf{S}^\top[j]$ is independently sampled from $\chi_k^m$, and the $j$th error vectors $\mathbf{E}_0^\top[j], \mathbf{E}_1^\top[j]$ are independently sampled from $\chi_e^{m\mathfrak{l}}$, and therefore for $\mathsf{SampPRF}$ query $x$,

$$\mathbf{u}_{x\setminus(i+1)}^T = \mathbf{B}_{x_i}^\top[\mathsf{qind}(x \setminus i)] = \mathbf{S}^\top[\mathsf{qind}(x \setminus i)]\mathbf{A}_{x_i} + \mathbf{E}_{x_i}^\top[\mathsf{qind}(x \setminus i)],$$

so

$$\mathcal{Y}[x]^\top = (\mathbf{S}^\top[\mathsf{qind}(x \setminus i)]\mathbf{A}_{x_i} + \mathbf{E}_{x_i}^\top[\mathsf{qind}(x \setminus i)]) \cdot \mathbf{B}_{x/(i+1)} + \mathbf{e}_{x/(i+1)}^\top + \mathbf{v}_{x,i}^\top$$

Therefore, in this case, comparing with (35), we see that $\mathcal{B}$ simulates to $\mathcal{A}$ the same view as in $\mathbf{Hybrid_{i,1}}$, with $\mathbf{k}_{x\setminus i}^\top := \mathbf{S}^\top[\mathsf{qind}(x \setminus i)]$.

On the other hand, in the case that the input MLWE instance comes from the 'Random' MLWE distribution ($b = 0$), then $\mathbf{u}_{x\setminus(i+1)}^T = \mathbf{B}_{x_i}^\top[\mathsf{qind}(x \setminus i)]$ are uniform in $R_q^{m\mathfrak{l}}$ and independent for distinct $(i+1)$-bit prefixes $x \setminus (i+1) \in \{0,1\}^{i+1}$. Therefore, in this case, $\mathcal{B}$ simulates to $\mathcal{A}$ the same view as in $\mathbf{Hybrid_{i,0}}$.

The above two observations establish the claim (36). Furthermore, observe that in this game $\mathbf{Hybrid_{i,1}}$, the $\mathsf{SampPRF}$ output has the form

$$\mathcal{Y}[x]^\top := \mathbf{u}_{x\setminus(i+1)}^\top \cdot \mathbf{B}_{x/(i+1)} + \mathbf{e}_{x/(i+1)}^\top + \mathbf{v}_{x,i}^\top \in R_q^{m\mathfrak{l}}, \tag{37}$$

which establishes the induction hypothesis (33) for $i+1$. Applying the induction $L-1$ times with $i = 1, \ldots, L-1$, we arrive at the final hybrid $\mathbf{Hybrid_{L-1,0}}$, in which the output of $\mathsf{SampPRF}$ on query $x$ is

$$\mathcal{Y}[x]^\top := \mathbf{u}_{x\setminus L}^\top \cdot \mathbf{B}_{x/L} + \mathbf{e}_{x/L}^\top + \mathbf{v}_{x,L-1}^\top \tag{38}$$

$$= \mathbf{u}_x^\top + \mathbf{v}_{x,L-1}^\top \in R_q^{m\mathfrak{l}}, \tag{39}$$

which is uniformly random in $R_q^{m\mathfrak{l}}$ and independent for distinct $x \in \{0,1\}^L$, since $\mathbf{u}_x^\top$ is uniformly random in $R_q^{m\mathfrak{l}}$ and independent of $\mathbf{v}_{x,L-1}^\top$ (where, on the second line of (38), we have used the fact that $x \setminus L := x$, $\mathbf{B}_{x/L} := \mathbf{I}$, and $\mathbf{e}_{x/L}^\top = \mathbf{0}$). Therefore, the view of $\mathcal{A}$ in $\mathbf{Hybrid_{L-1,0}}$ is identical to $\mathcal{A}$'s view in the random ($b = 0$) experiment $\mathsf{MLWE\text{-}PRF}^0$, as required. This completes the description of the $L$ pairs of hybrids.

Putting together the bounds (32), and (36), (34) for $i = 1 \ldots, L-1$, we get:

$$\mathsf{Adv}^{\mathsf{MLWE\text{-}PRF}}_{\mathcal{A}}(\lambda) = |\Pr[\mathbf{Hybrid_{L-1,0}} = 1] - \Pr[\mathbf{Hybrid_{0,1}} = 1]| \tag{40}$$

$$\leq \sum_{i=0}^{L-1} |\Pr[\mathbf{Hybrid_{i,1}} = 1] - \Pr[\mathbf{Hybrid_{i,0}} = 1]| \tag{41}$$

$$\leq \mathsf{Adv}^{\mathsf{MLWE}}_{\mathcal{B}}(\lambda) + \sum_{i=1}^{L-1} \mathsf{Adv}^{\mathsf{MLWE}}_{\mathcal{B}_i}(\lambda) \tag{42}$$

$$\leq \mathsf{Adv}^{\mathsf{MLWE}}_{\mathcal{B}}(\lambda) + (L-1) \cdot \mathsf{Adv}^{\mathsf{MLWE}}_{\overline{\mathcal{B}}}(\lambda), \tag{43}$$

$$\tag{44}$$

where $\overline{\mathcal{B}}$ denotes the adversary among $\mathcal{B}_1, \ldots, \mathcal{B}_{L-1}$ with the highest advantage. This completes the proof. □

### D.2 Correctness Analysis

The OPRF correctness analysis remains unaffected whether a part of the PRF input $x$ in the computation of $\mathsf{H}(x)$ is public or not. That's why we just write $\mathbf{B}_x := \mathsf{H}(x)$ in the correctness analysis to keep the notation a bit simpler.

**Definition 19.** *We say that function family* $\mathsf{H}$ *satisfies* $\epsilon_u$-uniformity *if, for each fixed* $x \in \{0,1\}^L$*, the distribution of any fixed* $\mathbb{Z}_q$ *coordinate of* $\mathbf{B}_x\mathbf{k} := \mathsf{H}(x)\mathbf{k} \in R_q^h$ *over the choice of* $\mathsf{H}$ *(in* $\mathsf{F.Setup}(1^\lambda)$*) and* $\mathbf{k} \xleftarrow{\$} \chi_k^m$ *is within statistical distance*

$$\epsilon_u \leq 2^{-(\kappa+2+\log(hd))} \tag{45}$$

*from the uniform distribution on* $\mathbb{Z}_q$*.*

**Lemma 7.** *Fix* $\kappa, h, d$*. Let* $B_f(\kappa, d)$ *denote an upper bound on a fixed coordinate of* $\mathbf{e}_f = \mathbf{e}'_s - \mathbf{R}\mathbf{e}_s$ *(as in (49)) that holds except with probability* $p_e \leq 2^{-(\kappa+2+\log(hd))}$*. Also, assume that the function family* $\mathsf{H}$ *satisfies* $\epsilon_u$-uniformity *(as per Definition 19) with*

$$\epsilon_u \leq 2^{-(\kappa+2+\log(hd))}. \tag{23}$$

*Then, for any fixed* $x \in \{0,1\}^L$*,* $2^{-\kappa}$*-correctness holds (as per Definition 2) if*

$$q/p \geq 2^{\kappa+2} \cdot hd \cdot (2B_f(\kappa, d) + 1). \tag{24}$$

*Proof.* Fix $(t, x)$, and let $E$ and $\delta$ denote respectively the event that a correctness error occurs and the correctness error probability. We have

$$\delta := \Pr[E] := \Pr[\,\lfloor \mathbf{u}_x - \mathbf{R}\mathbf{v}_k \rceil_p \neq \lfloor \mathbf{B}_x\mathbf{k} \rceil_p\,] \tag{46}$$

$$= \Pr[\,\lfloor \mathbf{B}_x\mathbf{k} + \mathbf{e}_f \rceil_p \neq \lfloor \mathbf{B}_x\mathbf{k} \rceil_p\,] \tag{47}$$

$$\tag{48}$$

where the final error $\mathbf{e}_f$ satisfies

$$\mathbf{e}_f := \mathbf{u}_x - \mathbf{R}\mathbf{v}_k - \mathbf{B}_x\mathbf{k} = \mathbf{e}'_s - \mathbf{R}\mathbf{e}_s. \tag{49}$$

For $i \in [h]$ and $j \in [d]$, let $\mathbf{e}_{f,i} \in R$ denote the $i$th ring element of $\mathbf{e}_f \in R^h$ and let $e_{f,i,j} \in \mathbb{Z}$ denote the $j$'th integer coefficient of $\mathbf{e}_{f,i}$ in the coefficient embedding of $R$. Similarly, viewing $\mathbf{B}_x$ in its representation over $\mathbb{Z}$, we denote by $\mathbf{b}^\top_{x,i,j} \in \mathbb{Z}_q^{md}$ the $j$'th row of the column rot (negacyclic) matrix corresponding to $\mathbf{B}_x$. Let $B_p \subset [0, q-1]$ denote the set of rounding mod $p$ interval boundary points, i.e. $B_p = \{\frac{q}{2p}, \frac{q}{2p} + \frac{q}{p}, \ldots, \frac{q}{2p} + (p-1)\cdot\frac{q}{2p}\}$. If event $E$ occurs, then there must exist some $i \in [h]$ and $j \in [d]$ such that $\mathbf{b}^\top_{x,i,j}\mathbf{k} \in \mathbb{Z}_q$ lands within distance $|e_{f,i,j}|$ of a rounding interval boundary point in $B_p$. Therefore, we have

$$\begin{aligned} \delta &\leq \Pr[\,\exists i \in [h], j \in [d] \text{ s.t. } \mathbf{b}^\top_{x,i,j}\mathbf{k} \in B_p \pm |e_{f,i,j}|\,] \\ &\leq \sum_{i\in[h], j\in[d]} \delta_{i,j}, \text{ where } \delta_{i,j} := \Pr[\,\mathbf{b}^\top_{x,i,j}\mathbf{k} \in B_p \pm |e_{f,i,j}|\,]. \end{aligned} \tag{50}$$

Now, for each fixed $i \in [h]$ and $j \in [d]$, we have

$$\begin{aligned}
\delta_{i,j} &\leq \Pr_{u \xleftarrow{\$} \mathbb{Z}_q}[\, u \in B_p \pm |e_{f,i,j}|\,] + \epsilon_u \\
&\leq \Pr_{u \xleftarrow{\$} \mathbb{Z}_q}[\, u \in B_p \pm B_f(\kappa, d)\,] + p_e + \epsilon_u \\
&= \frac{|(B_p \pm B_f(\kappa,d)) \cap \mathbb{Z}_q|}{|\mathbb{Z}_q|} + p_e + \epsilon_u \\
&\leq \frac{p \cdot (2B_f(\kappa,d)+1)}{q} + p_e + \epsilon_u \\
&\leq 2^{-\kappa}/(hd),
\end{aligned}$$

where the first inequality above uses the assumed $\epsilon_u$-uniformity of $\mathsf{H}$, the second inequality uses the bound $|e_{f,i,j}| \leq B_f(\kappa, d)$ that holds except with probability $\leq p_e$, the third inequality uses $|B_p| = p$ and the last inequality uses the assumed upper bounds $\leq 2^{-\kappa}/(3hd)$ on the three terms on the right-hand side of the third inequality. It follows that $\delta \leq \sum_{i\in[h],j\in[d]} \delta_{i,j} \leq 2^{-\kappa}$, as claimed. □

**Bounds for BLMR13 PRF Instantiation of $\mathsf{H}$.** For BLMR13 (see [SSS23] for the module instantiation), we have $\chi_k := \mathcal{U}(R_q)$, $\mathsf{H}^{\mathsf{blmr}}_{\mathbf{A}_0,\mathbf{A}_1}(\mathbf{x}) := \prod_{i=0}^{L-1} \mathbf{A}_{x_i}$, where we sample $\mathbf{A}_0, \mathbf{A}_1 \xleftarrow{\$} R^{m\times m}_{\mathsf{bin}}$ and restart if $\mathbf{A}_0$ or $\mathbf{A}_1$ are not invertible over $R_q$.

**Lemma 8.** *For BLMR13, the function family $\mathsf{H}^{\mathsf{blmr}}$ satisfies 0-uniformity. In particular, the distribution of $\mathbf{B}^{\mathsf{blmr}}_x \cdot \mathbf{k} := \mathsf{H}^{\mathsf{blmr}}_{\mathbf{A}_0,\mathbf{A}_1}(x) \cdot \mathbf{k} \in R^{m\times m}_q$ over the choice of $\mathsf{H}^{\mathsf{blmr}}_{\mathbf{A}_0,\mathbf{A}_1}$ and $\mathbf{k} \xleftarrow{\$} \chi^m_k$ is the uniform distribution on $R^{m\times m}_q$. The result carries over to the case $\mathsf{H}(x) := \mathbf{B}^{\mathsf{blmr}}_x[0]$ (as in $\mathsf{F.Setup}(1^\lambda)$).*

*Proof.* Follows immediately from the uniformly random distribution $\chi_k$ and the invertibility of the matrix $\mathbf{B}^{\mathsf{blmr}}_x = \mathsf{H}^{\mathsf{blmr}}_{\mathbf{A}_0,\mathbf{A}_1}(x) = \prod_{i=0}^{L-1} \mathbf{A}_{x_i}$ over $R_q$, thanks to the invertibility of $\mathbf{A}_0$ and $\mathbf{A}_1$ over $R_q$. The result carries over if we simply focus on the first coordinate of $\mathbf{B}^{\mathsf{blmr}}_x \cdot \mathbf{k}$, i.e., $\mathbf{B}^{\mathsf{blmr}}_x[0] \cdot \mathbf{k}$. □

**Bounds for BP14 PRF Instantiation of $\mathsf{H}$.** For BP14 with $\chi_k := \mathcal{U}(R_q)$, $\mathbf{B}_x = \mathsf{H}(\mathbf{x}) := (\mathbf{A}_{x_0}\mathbf{B}'_1)^\top \in R^{1\times m}_q$, and $\mathfrak{l} := \lceil \log_\gamma q \rceil$ with $\gamma \geq 2$, $\mathbf{B}_{L-1} := \mathbf{G}^{-1}(\mathbf{A}_{x_{L-1}}) \in R^{m\mathfrak{l}\times m\mathfrak{l}}_q$, $\mathbf{B}_i := \mathbf{G}^{-1}(\mathbf{A}_{x_i}\mathbf{B}_{i+1}) \in R^{m\mathfrak{l}\times m\mathfrak{l}}_q$ for $i = L-2, \ldots, 1$. $\mathbf{B}'_1$ consists of the leftmost $h$ columns of $\mathbf{B}_1$, and $\mathsf{F.Setup}(1^\lambda)$ samples $\mathbf{A}_0, \mathbf{A}_1 \xleftarrow{\$} R^{m\times m\mathfrak{l}}_q$.

**Lemma 9.** *Assume that $q$ is prime. If the (truncated) BP14 function family $\mathsf{H}$ does not satisfy $\epsilon_u$-uniformity for some non-negligible $\epsilon_u$, then there exists a poly-time 1-query adversary against the pseudorandomness of the BP14 PRF with non-negligible advantage $\geq (1-1/p)\cdot\epsilon_u$.*

*Proof.* By hypothesis, there exists $x \in \{0,1\}^L$, $i \in [h]$ and $j \in [d]$ such that the statistical distance $\epsilon$ between the distribution of $\mathbf{b}^\top_{x,i,j}\mathbf{k} \in \mathbb{Z}_q$ and $\mathcal{U}(\mathbb{Z}_q)$ is $\geq \epsilon_u$, where $\mathbf{b}^\top_{x,i,j}$ denotes the $j$'th row of the column rot (negacyclic) matrix representation of $\mathbf{B}_x$ over $\mathbb{Z}$ of the $i$'th row of $\mathbf{B}_x \in R^{h\times m}_q$, and similarly $\mathbf{k}$ is viewed over $\mathbb{Z}$ as the concatenation of ring element coefficient vectors. Let $\mathsf{Z}$ denote the event that $\mathbf{b}^\top_{x,i,j} = \mathbf{0}$. Then we have $\epsilon = \epsilon_{nz} \cdot (1 - \Pr[\mathsf{Z}]) + \epsilon_z \cdot \Pr[\mathsf{Z}]$, where $\epsilon_{nz}$ (resp. $\epsilon_z$) denotes the statistical distance between the distribution of $\mathbf{b}^\top_{x,i,j}\mathbf{k} \in \mathbb{Z}_q$ and $\mathcal{U}(\mathbb{Z}_q)$ conditioned on the event that $\mathsf{Z}$ does not occur (resp. $\mathsf{Z}$ occurs). Conditioned on $\mathsf{Z}$ not occuring, we have $\mathbf{b}^\top_{x,i,j} \neq \mathbf{0}$, and thus $\mathbf{b}^\top_{x,i,j}$ has a non-zero and hence (by primality of $q$) invertible component in $\mathbb{Z}_q$. Thanks to the independence and uniformity of $\mathbf{k}$'s components in $\mathbb{Z}_q$, it follows that in this case $\mathbf{b}^\top_{x,i,j}\mathbf{k}$ is uniformly distributed in $\mathbb{Z}_q$, so $\epsilon_{nz} = 0$. Therefore $\epsilon = \epsilon_z \cdot \Pr[\mathsf{Z}] \leq \Pr[\mathsf{Z}]$. It follows that $\Pr[\mathsf{Z}] \geq \epsilon \geq \epsilon_u$ is non-negligible. Then there exists a pseudorandomness adversary $\mathcal{A}$ against the BP14 PRF that works as follows: it queries $x$ to its oracle to get $\mathbf{z} \in R^h_p$. If $\mathsf{Z}$ occurs then $\mathcal{A}$ outputs 1 if $z_{i,j} = 0$ and output 0 else. If $\mathsf{Z}$ does not occur, $\mathcal{A}$ outputs a random coin. In the case that $\mathcal{A}$'s oracle is the BP14 PRF, if the event $\mathsf{Z}$ occurs, the oracle will output 0 with probability 1 and hence $\mathcal{A}$ will output 1 with probability $p_P = \Pr[\mathsf{Z}] + 1/2(1-\Pr[\mathsf{Z}])$. In the other case that $\mathcal{A}$'s oracle is a uniformly random function with range $R^h_p$, the event $z_{i,j} = 0$ will occur with probability $1/p$ independently of event $\mathsf{Z}$, and hence in this case

$\mathcal{A}$ will output 1 with probability $p_U = \Pr[\mathsf{Z}] \cdot 1/p + 1/2(1 - \Pr[\mathsf{Z}])$. It follows that the distinguishing advantage of $\mathcal{A}$ against the BP14 PRF security is lower bounded as $|p_P - p_U| \geq (1 - 1/p) \cdot \Pr[\mathsf{Z}] > (1 - 1/p) \cdot \epsilon_u$, as claimed. □

*Remark 7.* We note that the $\epsilon_u$-uniformity Lemma above assumes $\chi_k$ has uniformly random coordinates in $\mathbb{Z}_q$, but the BP14 PRF could also be instantiated with $\chi_k$ restricted to small coefficients. In the latter case, the $\epsilon_u$-uniformity seems more difficult to prove but we heuristically expect it to still be satisfied.

*Remark 8.* We note that our correctness analysis extends to a *malicious* server setting by relying on the fact that the NIZK proof $\pi_s$ by the server ensures that the error terms $(\mathbf{e}_s, \mathbf{e}'_s)$ have small coefficients (say bounded by $\beta$ and $\beta_1$, respectively). Particularly, using the worst-case bound in (25) for $B_f$ gives the correctness analysis requirement against a malicious server.

### D.3 Pseudorandomness Analysis

**Theorem 5.** *The POPRF construction* $\mathsf{F}$ *from Figures 8 and 9 satisfies pseudorandomness given in Definition 3, with random oracles* $\mathsf{RO}_r$ *and* $\mathsf{RO}_z$*, if:*

- *The client argument system* $\mathsf{NIZK}_1$ *is computationally sound and the server argument system* $\mathsf{NIZK}_2$ *is computationally zero-knowledge (Definition 18),*
- *The commitment scheme* $\mathsf{COM}$ *is computationally hiding and extractable (Definitions 15 and 17), and*
- *The* $\mathsf{MLWE\text{-}PRF\text{-}RU}_{\mathsf{param}}$ *assumption (Definition 10) holds for* $\mathsf{param} := (\mathsf{H}, Q, Q^{\infty}_{x,t}, Q_{\mathsf{M}}, q, m, \ell + m, h, L, \beta_r, \overrightarrow{\chi})$ *and* $\overrightarrow{\chi} := (\chi, \chi_1, \chi_k)$*, where* $Q$ *corresponds to the number of* $\mathsf{BlindEval}$ *queries,* $Q^{\infty}_{x,t}$ *to the maximum number of identical* $(x||t)$ *queries to* $\mathsf{BlindEval}$*, and* $Q_{\mathsf{M}}$ *to the number of* $\mathsf{RO}_r$ *queries, respectively.*

*More precisely, for any* PPT *adversary* $\mathcal{A}$*, there exist* PPT *adversaries* $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4, \mathcal{B}_5$ *and* $\mathcal{B}_6$ *against the computational zero-knowledge of* $\mathsf{NIZK}_2$*, computational soundness of* $\mathsf{NIZK}_1$*, hiding and extractability of* $\mathsf{COM}$*, and* $\mathsf{MLWE\text{-}PRF\text{-}RU}_{\mathsf{param}}$ *assumption, respectively, such that*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{po\text{-}prf}}_{\mathsf{F},\mathcal{A},\mathcal{S},\mathsf{RO}_r,\mathsf{RO}_z}(\lambda) \leq\ & \mathsf{Adv}^{\mathsf{CZK}}_{\mathsf{NIZK}_2,\mathcal{B}_1}(\lambda) + Q_{\mathsf{B}} \cdot \mathsf{Adv}^{\mathsf{CS}}_{\mathsf{NIZK}_1,\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathsf{CH}}_{\mathsf{COM},\mathcal{B}_3}(\lambda) \\ & + Q_{\mathsf{RO}_r} \cdot \mathsf{Adv}^{\mathsf{Ext}}_{\mathsf{COM},\mathcal{B}_4}(\lambda) + Q_{\mathsf{B}} \cdot \mathsf{Adv}^{\mathsf{Ext}}_{\mathsf{COM},\mathcal{B}_5}(\lambda) + \mathsf{Adv}^{\mathsf{iMLWE}_{\mathsf{param}}}_{\mathcal{B}_6}(\lambda) \\ & + Q_{\mathsf{RO}_z}\left(\frac{1}{p} + \frac{1}{q}\right)^{dh},\end{aligned}$$

*where* $Q_{\mathsf{B}}$*,* $Q_{\mathsf{RO}_r}$ *and* $Q_{\mathsf{RO}_z}$ *denote the number of* $\mathsf{BlindEval}$*,* $\mathsf{RO}_r$ *and* $\mathsf{RO}_z$ *queries, respectively, that the adversary* $\mathcal{A}$ *makes.*

*Proof.* We consider the POPRF pseudorandomness game given in Definition 3.

We note that the adversary $\mathcal{A}$ has access to oracles $\mathsf{Eval}, \mathsf{PreProcServer}, \mathsf{BlindEval}$ and $\mathsf{Prim}$, where $\mathsf{Prim}$ denotes the random oracles $\mathsf{RO}_r$ and $\mathsf{RO}_z$. The random oracles $\mathsf{RO}_r$ and $\mathsf{RO}_z$ internally keep lists $\mathcal{H}$ and $\mathcal{F}$, which are initially set to $\perp$. On input $x$ from the domain of $\mathsf{RO}_r$ (resp. $\mathsf{RO}_z$), the oracle first checks if $\mathcal{H}[x] \neq \perp$ (resp. $\mathcal{F}[x] \neq \perp$), i.e., checks if it has already been defined via a prior query. If it is, then it returns $\mathcal{H}[x]$ (resp. $\mathcal{F}[x]$), and otherwise, it sets $\mathcal{H}[x]$ (resp. $\mathcal{F}[x]$) to a uniformly random value in the range of $\mathsf{RO}_r$ (resp. $\mathsf{RO}_z$), and returns it. The same logic applies to the other internal lists, $\mathcal{Y}$ and $\mathcal{Z}$, used during the security proof.

The security proof follows in a series of hybrids, where we start with the pseudorandomness game $\mathsf{POPRF}^1_{\mathsf{F},\mathcal{A},\mathcal{S},\mathsf{RO}_r,\mathsf{RO}_z}$, i.e., $b = 1$, and we gradually apply changes until we end up with the game $\mathsf{POPRF}^0_{\mathsf{F},\mathcal{A},\mathcal{S},\mathsf{RO}_r,\mathsf{RO}_z}$, i.e., $b = 0$ and the oracle calls are answered by the simulator $\mathcal{S}$. Let $\mathsf{Adv}_{\mathcal{A},i}(\lambda)$ denote the advantage of $\mathcal{A}$ in $\mathbf{Hybrid_i}$. We consider the hybrids listed below, and for each hybrid we depict the changes using figures, and since the adversarially controlled algorithms are not changed, we do not include them in the figures.

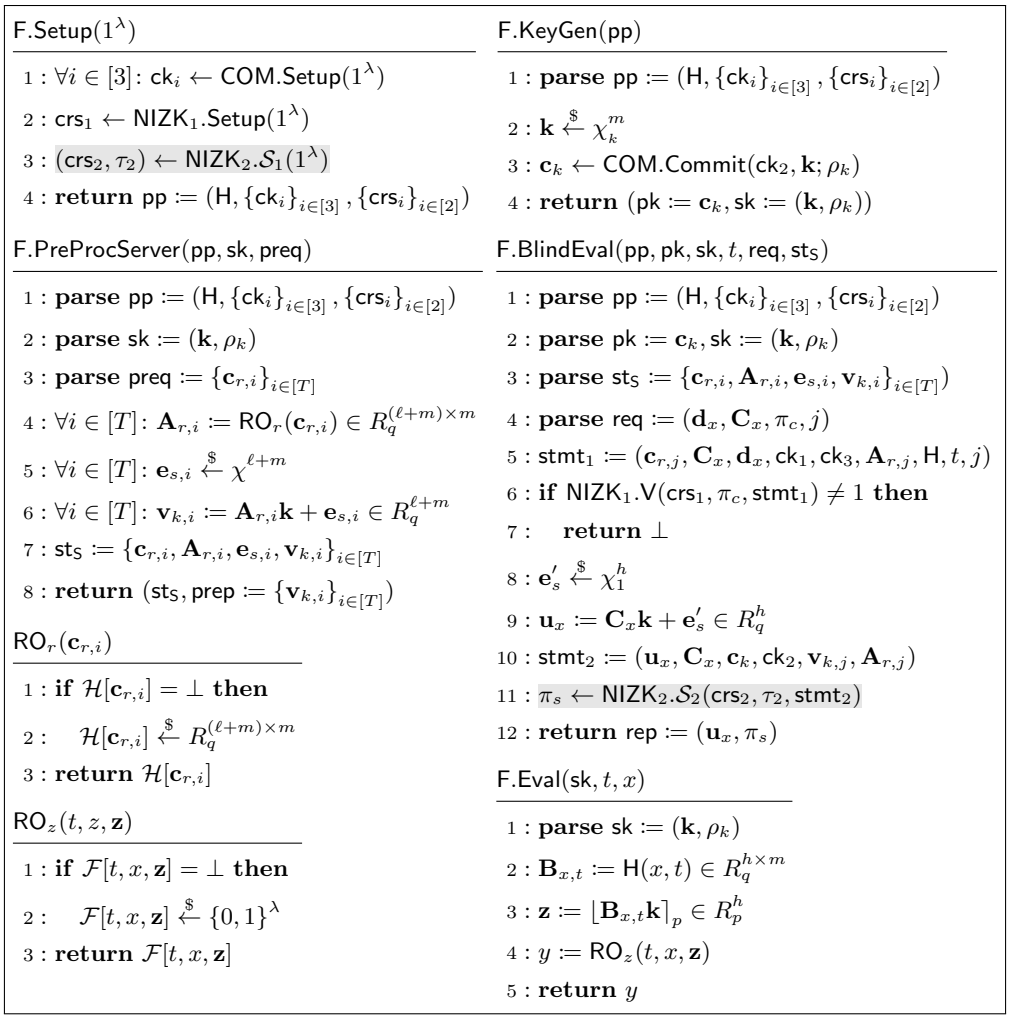


Fig. 15: **Hybrid$_1$** of pseudorandomness proof.

**Hybrid$_0$**: This corresponds to the pseudorandomness game $\mathsf{POPRF}^1_{\mathsf{F},\mathcal{A},\mathcal{S},\mathsf{RO}_r,\mathsf{RO}_z}$ (Definition 3), and the interaction follows as in Figures 8 and 9. Hence, it follows that

$$\mathsf{Adv}^{\text{po-prf}}_{\mathsf{F},\mathcal{A},\mathcal{S},\mathsf{RO}_r,\mathsf{RO}_z}(\lambda) = \mathsf{Adv}_{\mathcal{A},0}(\lambda).$$

**Hybrid$_1$**: In this hybrid, we replace $\mathsf{crs}_2$ and proof $\pi_s$ with a simulated CRS and proof $(\mathsf{crs}^*_2, \pi^*_s)$, as shown in Figure 15.

In order to argue indistinguishability between **Hybrid$_0$** and **Hybrid$_1$**, we construct a reduction $\mathcal{B}_1$ to the computational zero-knowledge property of $\mathsf{NIZK}_2$. We note that all queries are answered as in **Hybrid$_0$** with the exception of BlindEval queries. The only difference here is that $\mathcal{B}_1$ receives the CRS $\mathsf{crs}_2$ from its zero-knowledge challenger instead of generating it via the $\mathsf{NIZK}_2.\mathsf{Setup}$ algorithm. When $\mathcal{A}$ makes a BlindEval query, $\mathcal{B}_1$ proceeds as in **Hybrid$_0$** to compute $\mathbf{v}_{k,j}$ and $\mathbf{u}_x$, but it makes an oracle call to its zero-knowledge challenger with the input $((\mathbf{u}_x, \mathbf{C}_x, \mathbf{c}_k, \mathsf{ck}_2, \mathbf{v}_{k,j}, \mathbf{A}_{r,j}), (\mathbf{k}, \mathbf{e}_{s,j}, \mathbf{e}'_s, \rho_k))$ to obtain the proof $\pi_s$.

If the zero-knowledge challenger of $\mathsf{NIZK}_2$ used the honest setup and prover algorithms to generate $\mathsf{crs}_2$ and $\pi_s$, then we are exactly in **Hybrid$_0$**, and if it used the simulator, then we are in **Hybrid$_1$**. Therefore, if $\mathcal{A}$ can distinguish between the two hybrids with non-negligible advantage, then $\mathcal{B}_1$ can break the computational

zero-knowledge property of $\mathsf{NIZK}_2$. Hence, it follows that

$$\left|\mathsf{Adv}_{\mathcal{A},0}(\lambda) - \mathsf{Adv}_{\mathcal{A},1}(\lambda)\right| \leq \mathsf{Adv}^{\mathsf{CZK}}_{\mathsf{NIZK}_2,\mathcal{B}_1}(\lambda).$$

**Hybrid$_2$**: Let $\mathsf{BAD}_1$ be the event that for a blind evaluation query $(t, \mathsf{req} \coloneqq (\mathbf{d}_x, \mathbf{C}_x, \pi_c, j))$, it holds that $\mathsf{NIZK}_1.\mathsf{V}(\mathsf{crs}_1, \pi_c, (\mathbf{c}_{r,j}, \mathbf{C}_x, \mathbf{d}_x, \mathsf{ck}_1, \mathsf{ck}_3, \mathbf{A}_{r,j}, \mathsf{H}, t, j)) = 1$ (where $\mathbf{c}_{r,j}$ is in $\mathsf{st}_\mathsf{S}$), but $\nexists(\mathbf{R}_j, x, \rho_{r,j}, \rho_x)$ such that

$$\begin{aligned}&\mathbf{C}_x = \mathbf{R}_j\mathbf{A}_{r,j} + \mathsf{H}(x,t) \bmod q\\ \wedge\ &\mathbf{c}_{r,j} = \mathsf{COM.Commit}(\mathsf{ck}_3, \mathbf{R}_j; \rho_{r,j})\\ \wedge\ &\mathbf{d}_x = \mathsf{COM.Commit}(\mathsf{ck}_1, x; \rho_x).\end{aligned}$$

Next, we prove that the $\mathsf{BAD}_1$ event can only happen with negligible probability due to the soundness of $\mathsf{NIZK}_1$ argument system. Looking ahead, we need here such a reduction to the soundness of $\mathsf{NIZK}_1$ because later in **Hybrid$_4$** we do a reduction to the extractability of the commitment scheme, and there we need to ensure that the adversarial commitment $\mathbf{c}_{r,j}$ is well-formed.

Clearly, we have that **Hybrid$_1$** and **Hybrid$_2$** are identical until the event $\mathsf{BAD}_1$ happens. Hence, we show that we can bound the probability $\Pr[\mathsf{BAD}_1]$ by constructing a reduction $\mathcal{B}_2$ to the computational soundness of $\mathsf{NIZK}_1$ argument system. We note that all queries are answered as in **Hybrid$_1$**. Let $Q_\mathsf{B}$ denote the maximum number of $\mathsf{BlindEval}$ queries that $\mathcal{A}$ performs. $\mathcal{B}_2$ randomly chooses an index $i^* \in [Q_\mathsf{B}]$ to serve as its guess for the query where the proof $\pi_c$ will be forged by the adversary $\mathcal{A}$. For $\mathsf{req} \coloneqq (\mathbf{d}_x, \mathbf{C}_x, \pi_c, j)$ that corresponds to the $i^*$-th blind evaluation query, $\mathcal{B}_2$ simply outputs $\pi_c$. We emphasize that $\mathcal{B}_2$ avoids recovering any parts of the witness $(\mathbf{R}_j, x, \rho_{r,j}, \rho_x)$ (which are computationally infeasible to compute), and instead just guesses which of the proofs would have caused the event $\mathsf{BAD}_1$ to happen. Hence, we have that

$$\left|\mathsf{Adv}_{\mathcal{A},1}(\lambda) - \mathsf{Adv}_{\mathcal{A},2}(\lambda)\right| \leq Q_\mathsf{B} \cdot \mathsf{Adv}^{\mathsf{CS}}_{\mathsf{NIZK}_1,\mathcal{B}_2}(\lambda).$$

**Hybrid$_3$**: In this hybrid, during key generation we compute the commitment $\mathbf{c}_k \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_2, \mathbf{0}_m)$, where $\mathbf{0}_m$ is an all zero vector of length $m$, instead of computing a commitment ot the secret key $\mathbf{k}$ as in the previous hybrid. This change is shown in Figure 16.

In order to show indistinguishability between **Hybrid$_2$** and **Hybrid$_3$**, we construct a reduction $\mathcal{B}_3$ to the computational hiding property of $\mathsf{COM}$. For computing the public key $\mathsf{pk}$, $\mathcal{B}_3$ samples a random $\mathbf{k} \xleftarrow{\$} \chi_k^m$, sets $(m_0 \coloneqq \mathbf{k}, m_1 \coloneqq \mathbf{0}_m)$ and submits the pair $(m_0, m_1)$ to the computational hiding challenger of $\mathsf{COM}$, which responds with $\mathbf{c}_k$. At this point $\mathcal{B}_3$ sets $\mathsf{pk} \coloneqq \mathbf{c}_k$, and proceeds to answer the oracle queries as in the previous hybrid.

If the computational hiding challenger of $\mathsf{COM}$ provided a commitment to $m_0$, then we are exactly in **Hybrid$_2$**, and if it provided a commitment to $m_1$, then we are in **Hybrid$_3$**. Therefore, if $\mathcal{A}$ can distinguish between the two hybrids with non-negligible advantage, then $\mathcal{B}_3$ can break the computational hiding property of $\mathsf{COM}$. Hence, it follows that

$$\left|\mathsf{Adv}_{\mathcal{A},2}(\lambda) - \mathsf{Adv}_{\mathcal{A},3}(\lambda)\right| \leq \mathsf{Adv}^{\mathsf{CH}}_{\mathsf{COM},\mathcal{B}_3}(\lambda).$$

**Hybrid$_4$**: In this hybrid, we replace the commitment key $\mathsf{ck}_3$ with a trapdoor variant $(\mathsf{ck}_3, \mathsf{td}_3) \leftarrow \mathsf{COM}.\mathcal{E}_1(1^\lambda)$, and use the trapdoor $\mathsf{td}_3$ to extract the committed values queried to $\mathsf{RO}_r$ oracle, i.e., run $\mathbf{R}_i \leftarrow \mathsf{COM}.\mathcal{E}_2(\mathsf{ck}_3, \mathsf{td}_3, \mathbf{c}_{r,i})$, and store the $\mathbf{R}_i$ in the list $\mathcal{R}$ for later use, as depicted in Figure 17.

In order to prove indistinguishability between **Hybrid$_3$** and **Hybrid$_4$** we construct an adversary $\mathcal{B}_4$ against the extractability property of $\mathsf{COM}$. We note that all queries are answered as in **Hybrid$_3$** with the exception of $\mathsf{RO}_r$ queries. $\mathcal{B}_4$ samples a trapdoor commitment key $(\mathsf{ck}_3, \mathsf{td}_3) \leftarrow \mathsf{COM}.\mathcal{E}_1(1^\lambda)$, instead of generating $\mathsf{ck}_3$ via the $\mathsf{COM.Setup}$ algorithm. Let $Q_{\mathsf{RO}_r}$ denote the number of $\mathsf{RO}_r$ queries that the adversary $\mathcal{A}$ makes. For each such query $\mathbf{c}_{r,i}$, $\mathcal{B}_4$ uses the trapdoor $\mathsf{td}$ to extract $\mathbf{R}_i \leftarrow \mathsf{COM}.\mathcal{E}_2(\mathsf{ck}_3, \mathsf{td}_3, \mathbf{c}_{r,i})$ and stores it in the list $\mathcal{R}[\mathbf{c}_{r,i}] \coloneqq \mathbf{R}_i$. We remark that these $\mathbf{R}_i$ values will be used at a later hybrid.

Due to the soundness of the client proof $\pi_c$ (as proven in **Hybrid$_2$**) we know that the adversarial commitment $\mathbf{c}_{r,i}$ is well-formed, and hence, the extractor extracts the correct $\mathbf{R}_i$. Hence, if $\mathcal{A}$ can distinguish

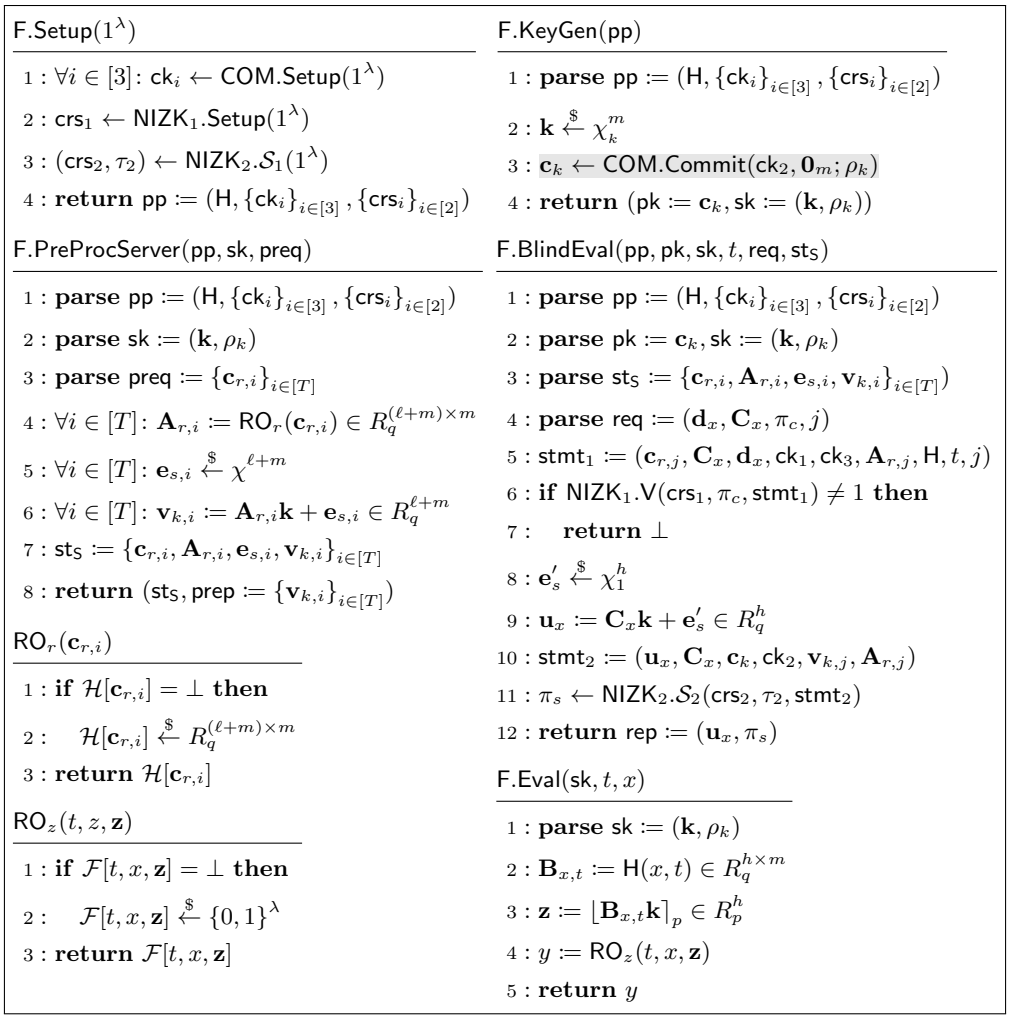


Fig. 16: **Hybrid$_3$** of pseudorandomness proof.

between the two hybrids with non-negligible advantage, then $\mathcal{B}_4$ can break the extractability property of COM. Moreover, since the adversary $\mathcal{A}$ makes at most $Q_{\mathsf{RO}_r}$ queries to $\mathsf{RO}_r$ oracle, it follows that

$$\left|\mathsf{Adv}_{\mathcal{A},3}(\lambda) - \mathsf{Adv}_{\mathcal{A},4}(\lambda)\right| \leq Q_{\mathsf{RO}_r} \cdot \mathsf{Adv}^{\mathsf{Ext}}_{\mathsf{COM},\mathcal{B}_4}(\lambda).$$

**Hybrid$_5$**: In this hybrid, we replace the commitment key $\mathsf{ck}_1$ with a trapdoor variant $(\mathsf{ck}_1, \mathsf{td}_1) \leftarrow \mathsf{COM}.\mathcal{E}_1(1^\lambda)$, and use the trapdoor $\mathsf{td}_1$ to extract $x \leftarrow \mathsf{COM}.\mathcal{E}_2(\mathsf{ck}_1, \mathsf{td}_1, \mathbf{d}_x)$ when receiving BlindEval queries, as depicted in Figure 18. We note that the extracted input $x$ will be used in the next hybrid.

The indistinguishability between **Hybrid$_4$** and **Hybrid$_5$** follows from the extractability property of COM as in the previous hybrid. Concretely, due to the soundness of the client proof $\pi_c$ (as proven in **Hybrid$_2$**) we know that the adversarial commitment $\mathbf{d}_x$ is well-formed, and hence, the extractor extracts the correct input $x$. Moreover, since the adversary $\mathcal{A}$ makes at most $Q_{\mathsf{B}}$ queries to the BlindEval oracle, it follows that

$$\left|\mathsf{Adv}_{\mathcal{A},4}(\lambda) - \mathsf{Adv}_{\mathcal{A},5}(\lambda)\right| \leq Q_{\mathsf{B}} \cdot \mathsf{Adv}^{\mathsf{Ext}}_{\mathsf{COM},\mathcal{B}_5}(\lambda),$$

where $\mathcal{B}_5$ is the reduction to the computational hiding of COM.

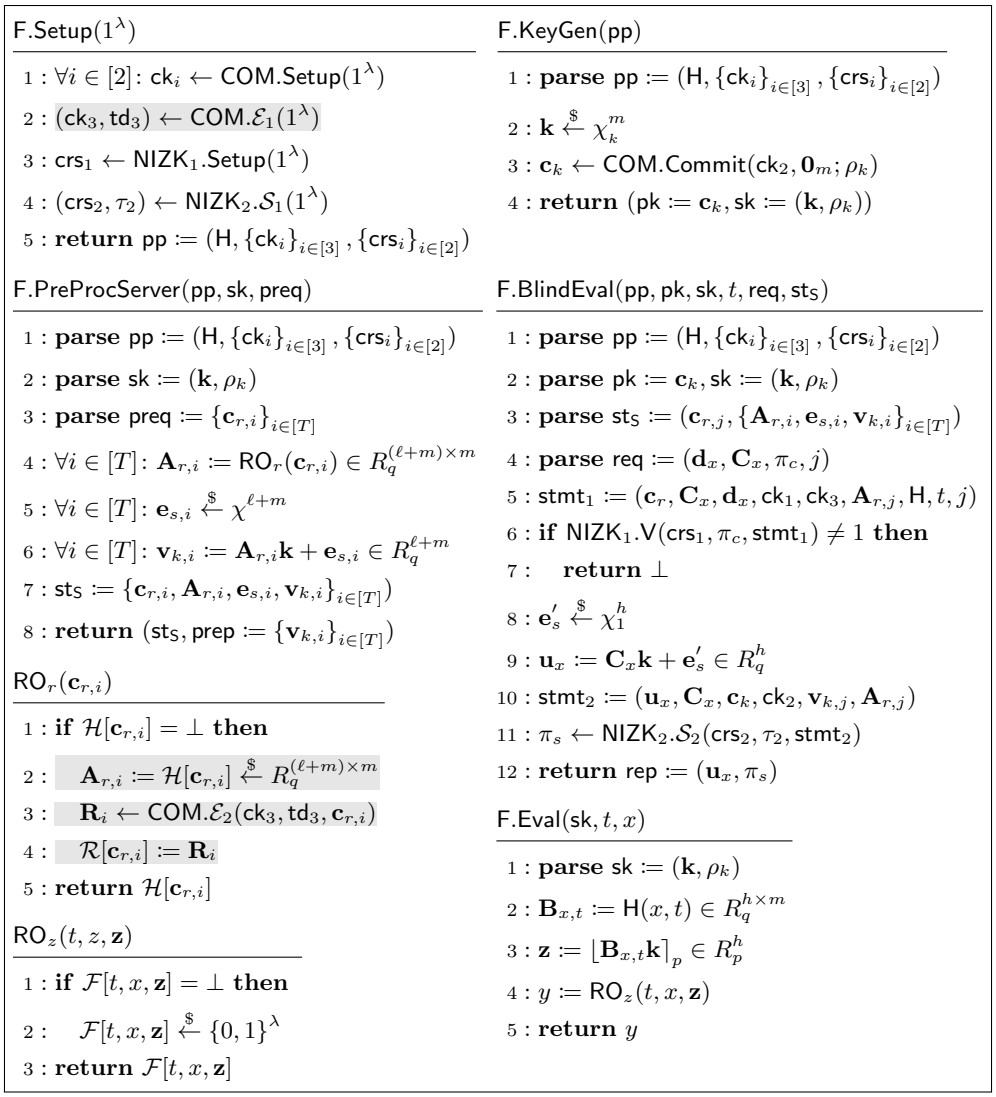


Fig. 17: **Hybrid$_4$** of pseudorandomness proof.

**Hybrid$_6$**: In this hybrid, we remove the secret key $\mathbf{k}$ and change the way we respond to the PreProcServer, BlindEval and Eval oracle queries. Concretely, for answering Eval queries, instead of computing $\mathbf{z} := \lfloor \mathbf{B}_{x,t}\mathbf{k} \rceil_p$, we sample a uniform $\mathcal{Z}[\mathbf{B}_{x,t}] \xleftarrow{\$} R_q^h$ and return $\mathbf{z} := \lfloor \mathcal{Z}[\mathbf{B}_{x,t}] \rceil_p$. For answering PreProcServer queries, instead of computing $\mathbf{v}_{k,i} := \mathbf{A}_{r,i}\mathbf{k} + \mathbf{e}_{s,i}$ we sample a uniformly random $\mathbf{v}_{k,i} \xleftarrow{\$} R_q^{\ell+m}$. Additionally, when answering the BlindEval queries, instead of computing $\mathbf{u}_x := \mathbf{C}_x\mathbf{k} + \mathbf{e}'_s$ we compute $\mathbf{u}_x := \mathbf{R}_j\mathbf{v}_{k,j} + \mathcal{Z}[\mathbf{B}_{x,t}] + \tilde{\mathbf{e}}' - \mathbf{R}_j\tilde{\mathbf{e}}$, for some $\tilde{\mathbf{e}}' \xleftarrow{\$} \chi_1^h$ and $\tilde{\mathbf{e}} \xleftarrow{\$} \chi^{\ell+m}$. These changes are depicted in Figure 19.

In order to show indistinguishability between **Hybrid$_5$** and **Hybrid$_6$** we construct an adversary $\mathcal{B}_6$ against the MLWE-PRF-RU$_{\mathsf{param}}$ assumption, for $\mathsf{param} := (\mathsf{H}, Q, Q^\infty_{x,t}, Q_\mathsf{M}, q, m, \ell+m, h, L, \beta_r, \overrightarrow{\chi})$, for $\overrightarrow{\chi} := (\chi, \chi_1, \chi_k)$. For answering $\mathsf{RO}_r$ query $(\mathbf{c}_{r,i})$, $\mathcal{B}_6$ extracts $\mathbf{R}_i \leftarrow \mathsf{COM}.\mathcal{E}_2(\mathsf{ck}_3, \mathsf{td}_3, \mathbf{c}_{r,i})$ (as in the previous hybrid), then queries the MLWE-PRF-RU oracle $(\mathbf{A}, \mathbf{c}) \leftarrow \mathsf{SampMLWEcom}(\mathbf{R}_i)$, and sets $\mathcal{H}[\mathbf{c}_{r,i}] := \mathbf{A}$ and $\mathbf{v}_{k,i} := \mathbf{c}$ (this is without loss of generality since when answering the PreProcServer oracle queries $\mathcal{B}_6$ will make a call to the $\mathsf{RO}_r$ oracle). Upon receiving a BlindEval query $(i, t, (\mathbf{d}_x, \mathbf{C}_x, \pi_c, j))$, $\mathcal{B}_6$ extracts $x \leftarrow$

$\mathsf{F.Setup}(1^\lambda)$

$(\mathsf{ck}_1, \mathsf{td}_1) \leftarrow \mathsf{COM}.\mathcal{E}_1(1^\lambda)$
$\mathsf{ck}_2 \leftarrow \mathsf{COM.Setup}(1^\lambda)$
$(\mathsf{ck}_3, \mathsf{td}_3) \leftarrow \mathsf{COM}.\mathcal{E}_1(1^\lambda)$
$\mathsf{crs}_1 \leftarrow \mathsf{NIZK}_1.\mathsf{Setup}(1^\lambda)$
$(\mathsf{crs}_2, \tau_2) \leftarrow \mathsf{NIZK}_2.\mathcal{S}_1(1^\lambda)$
**return** $\mathsf{pp} \coloneqq (\mathsf{H}, \{\mathsf{ck}_i\}_{i\in[3]}, \{\mathsf{crs}_i\}_{i\in[2]})$

$\mathsf{F.KeyGen}(\mathsf{pp})$

**parse** $\mathsf{pp} \coloneqq (\mathsf{H}, \{\mathsf{ck}_i\}_{i\in[3]}, \{\mathsf{crs}_i\}_{i\in[2]})$
$\mathbf{k} \xleftarrow{\$} \chi_k^m$
$\mathbf{c}_k \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_2, \mathbf{0}_m; \rho_k)$
**return** $(\mathsf{pk} \coloneqq \mathbf{c}_k, \mathsf{sk} \coloneqq (\mathbf{k}, \rho_k))$

$\mathsf{F.PreProcServer}(\mathsf{pp}, \mathsf{sk}, \mathsf{preq})$

**parse** $\mathsf{pp} \coloneqq (\mathsf{H}, \{\mathsf{ck}_i\}_{i\in[3]}, \{\mathsf{crs}_i\}_{i\in[2]})$
**parse** $\mathsf{sk} \coloneqq (\mathbf{k}, \rho_k)$
**parse** $\mathsf{preq} \coloneqq \{\mathbf{c}_{r,i}\}_{i\in[T]}$
$\forall i \in [T]\colon \mathbf{A}_{r,i} \coloneqq \mathsf{RO}_r(\mathbf{c}_{r,i}) \in R_q^{(\ell+m)\times m}$
$\forall i \in [T]\colon \mathbf{e}_{s,i} \xleftarrow{\$} \chi^{\ell+m}$
$\forall i \in [T]\colon \mathbf{v}_{k,i} \coloneqq \mathbf{A}_{r,i}\mathbf{k} + \mathbf{e}_{s,i} \in R_q^{\ell+m}$
$\mathsf{st_S} \coloneqq \{\mathbf{c}_{r,i}, \mathbf{A}_{r,i}, \mathbf{e}_{s,i}, \mathbf{v}_{k,i}\}_{i\in[T]}$
**return** $(\mathsf{st_S}, \mathsf{prep} \coloneqq \{\mathbf{v}_{k,i}\}_{i\in[T]})$

$\mathsf{RO}_r(\mathbf{c}_{r,i})$

**if** $\mathcal{H}[\mathbf{c}_{r,i}] = \bot$ **then**
  $\mathbf{A}_{r,i} \coloneqq \mathcal{H}[\mathbf{c}_{r,i}] \xleftarrow{\$} R_q^{(\ell+m)\times m}$
  $\mathbf{R}_i \leftarrow \mathsf{COM}.\mathcal{E}_2(\mathsf{ck}_3, \mathsf{td}_3, \mathbf{c}_{r,i})$
  $\mathcal{R}[\mathbf{c}_{r,i}] \coloneqq \mathbf{R}_i$
**return** $\mathcal{H}[\mathbf{c}_{r,i}]$

$\mathsf{RO}_z(t, z, \mathbf{z})$

**if** $\mathcal{F}[t, x, \mathbf{z}] = \bot$ **then**
  $\mathcal{F}[t, x, \mathbf{z}] \xleftarrow{\$} \{0,1\}^\lambda$
**return** $\mathcal{F}[t, x, \mathbf{z}]$

$\mathsf{F.BlindEval}(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}, t, \mathsf{req}, \mathsf{st_S})$

**parse** $\mathsf{pp} \coloneqq (\mathsf{H}, \{\mathsf{ck}_i\}_{i\in[3]}, \{\mathsf{crs}_i\}_{i\in[2]})$
**parse** $\mathsf{pk} \coloneqq \mathbf{c}_k, \mathsf{sk} \coloneqq (\mathbf{k}, \rho_k)$
**parse** $\mathsf{st_S} \coloneqq (\mathbf{c}_{r,j}, \{\mathbf{A}_{r,i}, \mathbf{e}_{s,i}, \mathbf{v}_{k,i}\}_{i\in[T]})$
**parse** $\mathsf{req} \coloneqq (\mathbf{d}_x, \mathbf{C}_x, \pi_c, j)$
$\mathsf{stmt}_1 \coloneqq (\mathbf{c}_r, \mathbf{C}_x, \mathbf{d}_x, \mathsf{ck}_1, \mathsf{ck}_3, \mathbf{A}_{r,j}, \mathsf{H}, t, j)$
**if** $\mathsf{NIZK}_1.\mathsf{V}(\mathsf{crs}_1, \pi_c, \mathsf{stmt}_1) \neq 1$ **then**
  **return** $\bot$
$x \leftarrow \mathsf{COM}.\mathcal{E}_2(\mathsf{ck}_1, \mathsf{td}_1, \mathbf{d}_x)$
$\mathbf{e}'_s \xleftarrow{\$} \chi_1^h$
$\mathbf{u}_x \coloneqq \mathbf{C}_x\mathbf{k} + \mathbf{e}'_s \in R_q^h$
$\mathsf{stmt}_2 \coloneqq (\mathbf{u}_x, \mathbf{C}_x, \mathbf{c}_k, \mathsf{ck}_2, \mathbf{v}_{k,j}, \mathbf{A}_{r,j})$
$\pi_s \leftarrow \mathsf{NIZK}_2.\mathcal{S}_2(\mathsf{crs}_2, \tau_2, \mathsf{stmt}_2)$
**return** $\mathsf{rep} \coloneqq (\mathbf{u}_x, \pi_s)$

$\mathsf{F.Eval}(\mathsf{sk}, t, x)$

**parse** $\mathsf{sk} \coloneqq (\mathbf{k}, \rho_k)$
$\mathbf{B}_{x,t} \coloneqq \mathsf{H}(x, t) \in R_q^{h\times m}$
$\mathbf{z} \coloneqq \lfloor \mathbf{B}_{x,t}\mathbf{k} \rceil_p \in R_p^h$
$y \coloneqq \mathsf{RO}_z(t, x, \mathbf{z})$
**return** $y$

Fig. 18: **Hybrid$_5$** of pseudorandomness proof.

$\mathsf{COM}.\mathcal{E}_2(\mathsf{ck}_1, \mathsf{td}_1, \mathbf{d}_x)$ (as in the previous hybrid) and makes an oracle call $\mathbf{z} \leftarrow \mathsf{SampPRF\text{-}RU}(\mathcal{H}[\mathbf{c}_r, j], x||t)$ to the MLWE-PRF-RU challenger. Then, $\mathcal{B}_6$ sets $\mathbf{u}_x \coloneqq \mathbf{R}_j\mathbf{v}_{k,j} + \mathbf{z}$. The rest of the queries are answered as in the previous hybrid.

If the MLWE-PRF-RU challenger used $b = 1$, then the view of $\mathcal{A}$ simulated by $\mathcal{B}_6$ is exactly as in **Hybrid$_5$**, and if it used $b = 0$, then the view of $\mathcal{A}$ is as in **Hybrid$_6$**. Therefore, if $\mathcal{A}$ can distinguish between the two hybrids with non-negligible advantage, then $\mathcal{B}_6$ can break the $\mathsf{MLWE\text{-}PRF\text{-}RU}_{\mathsf{param}}$ assumption, where $Q$ in the MLWE-PRF-RU assumption denotes the total number of BlindEval queries, $Q^\infty_{x,t}$ denotes the maximum number of identical $(x||t)$ queries and $Q_{\mathsf{M}}$ denotes the total number of $\mathsf{RO}_r$ queries, respectively, that the adversary $\mathcal{A}$ makes. Combined with the above claim we get that

$$\left|\mathsf{Adv}_{\mathcal{A},5}(\lambda) - \mathsf{Adv}_{\mathcal{A},6}(\lambda)\right| \leq \mathsf{Adv}^{\mathsf{iMLWE}_{\mathsf{param}}}_{\mathcal{B}_6}(\lambda).$$

**Hybrid$_7$**: In this hybrid we replace the $\mathsf{Eval}$ algorithm with a random oracle, which is depicted in Figure 20. This corresponds to the pseudorandomness game $\mathsf{POPRF}^0_{\mathsf{F},\mathcal{A},\mathcal{S},\mathsf{RO}_r,\mathsf{RO}_z}$ (Definition 3).

Let $\mathsf{BAD}_2$ be the event that the adversary $\mathcal{A}$ queries $(t, x, \mathbf{z})$ to $\mathsf{RO}_z$ oracle, for some $\mathbf{z} \coloneqq \lfloor \mathcal{Z}[t, x] \rceil_p$, before ever making a query to the $\mathsf{RO}_r$ oracle. Clearly, **Hybrid$_6$** and **Hybrid$_7$** are identical until the event $\mathsf{BAD}_2$ happens. Hence, we observe that the only distinguishing advantage the adversary $\mathcal{A}$ has between **Hybrid$_6$** and **Hybrid$_7$** is if $\mathcal{A}$ queries $\mathsf{RO}_z$ oracle with some tuple $(t, x, \mathbf{z})$, where initially it holds that $\mathbf{z} \neq \lfloor \mathcal{Z}[\mathbf{B}_{x,t}] \rceil_p$ (this covers the case when $\mathcal{Z}[\mathbf{B}_{x,t}] = \perp$), but after making a call to the $\mathsf{BlindEval}$ oracle, we have that it is set to correct $\mathcal{Z}[\mathbf{B}_{x,t}]$ such that $\mathbf{z} \coloneqq \lfloor \mathcal{Z}[\mathbf{B}_{x,t}] \rceil_p$, then $\mathcal{A}$ will obtain two different outputs for the same $(t, x, \mathbf{z})$ input to $\mathsf{RO}_z$ oracle (i.e., before and after making a call to the $\mathsf{BlindEval}$ oracle), and therefore, it can distinguish the hybrids. The probability of this happening corresponds to the aforementioned $\mathsf{BAD}_2$ event happening, which we now argue that it can only happen with a negligible probability.

Note that inside the $\mathsf{PreProcServer}$ oracle, we sample $\mathbf{v}_{k,j}$ uniformly from $R_q^{\ell+m}$, and inside the $\mathsf{BlindEval}$ oracle we sample $\mathcal{Z}[\mathbf{B}_{x,t}]$ uniformly from $R_q^h$. Then, we compute $\mathbf{u}_x \coloneqq \mathbf{R}_j \mathbf{v}_{k,j} + \mathcal{Z}[\mathbf{B}_{x,t}] + \tilde{\mathbf{e}}' - \mathbf{R}_j \tilde{\mathbf{e}} \in R_q^h$. Therefore, the only value that is under the control of the adversary is $\mathbf{R}_j$, and since the rest of the values are chosen uniformly randomly, we have that $\mathbf{u}_x$ is distributed uniformly over $R_q^h$.

Concretely, we have that the adversarially controlled term of $\mathbf{R}_j \mathbf{v}_{k,j}$ will actually cancel out during the computation of $\lfloor \mathbf{u}_x - \mathbf{R}_j \mathbf{v}_{k,j} d \rceil_p$ (in the $\mathsf{Finalize}$ algorithm). Since $\mathcal{Z}[\mathbf{B}_{x,t}]$ is sampled independently of $\tilde{\mathbf{e}}' - \mathbf{R}_j \tilde{\mathbf{e}}$ and after $\mathbf{R}_j$ is fixed by the adversary, the distribution of each coefficient of $\mathcal{Z}[\mathbf{B}_{x,t}] + \tilde{\mathbf{e}}' - \mathbf{R}_j \tilde{\mathbf{e}}$ will be uniform mod $q$. Then, the chance that the adversary can hit the correct value for a coefficient after rounding is at most $1/p + 1/q$ (the $1/q$ terms appears when $q$ is not a multiple of $p$). Since $\mathcal{A}$ is a $\mathsf{PPT}$ algorithm, it can only make polynomially many queries to $\mathsf{RO}_z$ oracle. Let $Q_{\mathsf{RO}_z}$ denote the total number of queries that $\mathcal{A}$ makes to $\mathsf{RO}_z$ oracle, then

$$\Pr\left[\mathsf{BAD}_2\right] \leq Q_{\mathsf{RO}_z} \cdot \left(\frac{1}{p} + \frac{1}{q}\right)^{dh} \eqqcolon \epsilon_{\mathsf{BAD}}.$$

Finally, we note that the simulator $\mathcal{S}$ answers the oracle calls as in **Hybrid$_7$**, and putting everything together, we obtain

$$\begin{aligned}\mathsf{Adv}^{\mathsf{po\text{-}prf}}_{\mathsf{F},\mathcal{A},\mathcal{S},\mathsf{RO}_r,\mathsf{RO}_z}(\lambda) \leq\ & \mathsf{Adv}^{\mathsf{CZK}}_{\mathsf{NIZK}_2,\mathcal{B}_1}(\lambda) + Q_{\mathsf{B}} \cdot \mathsf{Adv}^{\mathsf{CS}}_{\mathsf{NIZK}_1,\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathsf{CH}}_{\mathsf{COM},\mathcal{B}_3}(\lambda) \\ & + Q_{\mathsf{RO}_r} \cdot \mathsf{Adv}^{\mathsf{Ext}}_{\mathsf{COM},\mathcal{B}_4}(\lambda) + Q_{\mathsf{B}} \cdot \mathsf{Adv}^{\mathsf{Ext}}_{\mathsf{COM},\mathcal{B}_5}(\lambda) + \mathsf{Adv}^{\mathsf{iMLWE}_{\mathsf{param}}}_{\mathcal{B}_6}(\lambda) \\ & + \epsilon_{\mathsf{BAD}},\end{aligned}$$

as claimed. This completes the proof of Theorem 5. □

$\mathsf{F.Setup}(1^\lambda)$

$(\mathsf{ck}_1, \mathsf{td}_1) \leftarrow \mathsf{COM}.\mathcal{E}_1(1^\lambda)$

$\mathsf{ck}_2 \leftarrow \mathsf{COM.Setup}(1^\lambda)$

$(\mathsf{ck}_3, \mathsf{td}_3) \leftarrow \mathsf{COM}.\mathcal{E}_1(1^\lambda)$

$\mathsf{crs}_1 \leftarrow \mathsf{NIZK}_1.\mathsf{Setup}(1^\lambda)$

$(\mathsf{crs}_2, \tau_2) \leftarrow \mathsf{NIZK}_2.\mathcal{S}_1(1^\lambda)$

**return** $\mathsf{pp} \coloneqq (\mathsf{H}, \{\mathsf{ck}_i\}_{i\in[3]}, \{\mathsf{crs}_i\}_{i\in[2]})$

$\mathsf{F.KeyGen}(\mathsf{pp})$

**parse** $\mathsf{pp} \coloneqq (\mathsf{H}, \{\mathsf{ck}_i\}_{i\in[3]}, \{\mathsf{crs}_i\}_{i\in[2]})$

$\mathbf{c}_k \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_2, \mathbf{0}_m; \rho_k)$

**return** $(\mathsf{pk} \coloneqq \mathbf{c}_k, \mathsf{sk} \coloneqq \bot)$

$\mathsf{F.PreProcServer}(\mathsf{pp}, \mathsf{sk}, \mathsf{preq})$

**parse** $\mathsf{pp} \coloneqq (\mathsf{H}, \{\mathsf{ck}_i\}_{i\in[3]}, \{\mathsf{crs}_i\}_{i\in[2]})$

**parse** $\mathsf{sk} \coloneqq (\mathbf{k}, \rho_k)$

**parse** $\mathsf{preq} \coloneqq \{\mathbf{c}_{r,i}\}_{i\in[T]}$

$\forall i \in [T]\colon \mathbf{A}_{r,i} \coloneqq \mathsf{RO}_r(\mathbf{c}_{r,i}) \in R_q^{(\ell+m)\times m}$

$\forall i \in [T]\colon \mathbf{v}_{k,i} \xleftarrow{\$} R_q^{\ell+m}$

$\mathsf{st_S} \coloneqq \{\mathbf{c}_{r,i}, \mathbf{A}_{r,i}, \mathbf{v}_{k,i}\}_{i\in[T]}$

**return** $(\mathsf{st_S}, \mathsf{prep} \coloneqq \{\mathbf{v}_{k,i}\}_{i\in[T]})$

$\mathsf{RO}_r(\mathbf{c}_{r,i})$

**if** $\mathcal{H}[\mathbf{c}_{r,i}] = \bot$ **then**

  $\mathbf{A}_{r,i} \coloneqq \mathcal{H}[\mathbf{c}_{r,i}] \xleftarrow{\$} R_q^{(\ell+m)\times m}$

  $\mathbf{R}_i \leftarrow \mathsf{COM}.\mathcal{E}_2(\mathsf{ck}_3, \mathsf{td}_3, \mathbf{c}_{r,i})$

  $\mathcal{R}[\mathbf{c}_{r,i}] \coloneqq \mathbf{R}_i$

**return** $\mathcal{H}[\mathbf{c}_{r,i}]$

$\mathsf{RO}_z(t, z, \mathbf{z})$

**if** $\mathcal{F}[t, x, \mathbf{z}] = \bot$ **then**

  $\mathcal{F}[t, x, \mathbf{z}] \xleftarrow{\$} \{0,1\}^\lambda$

**return** $\mathcal{F}[t, x, \mathbf{z}]$

$\mathsf{F.BlindEval}(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}, t, \mathsf{req}, \mathsf{st_S})$

**parse** $\mathsf{pp} \coloneqq (\mathsf{H}, \{\mathsf{ck}_i\}_{i\in[3]}, \{\mathsf{crs}_i\}_{i\in[2]})$

**parse** $\mathsf{pk} \coloneqq \mathbf{c}_k, \mathsf{sk} \coloneqq (\mathbf{k}, \rho_k)$

**parse** $\mathsf{st_S} \coloneqq \{\mathbf{c}_{r,i}, \mathbf{A}_{r,i}, \mathbf{v}_{k,i}\}_{i\in[T]}$

**parse** $\mathsf{req} \coloneqq (\mathbf{d}_x, \mathbf{C}_x, \pi_c, j)$

$\mathsf{stmt}_1 \coloneqq (\mathbf{c}_{r,j}, \mathbf{C}_x, \mathbf{d}_x, \mathsf{ck}_1, \mathsf{ck}_3, \mathbf{A}_{r,j}, \mathsf{H}, t, j)$

**if** $\mathsf{NIZK}_1.\mathsf{V}(\mathsf{crs}_1, \pi_c, \mathsf{stmt}_1) \neq 1$ **then**

  **return** $\bot$

$x \leftarrow \mathsf{COM}.\mathcal{E}_2(\mathsf{ck}_1, \mathsf{td}_1, \mathbf{d}_x)$

$\mathbf{R}_j \coloneqq \mathcal{R}[\mathbf{c}_{r,j}]$

$\mathbf{B}_{x,t} \coloneqq \mathbf{C}_x - \mathbf{R}_j\mathbf{A}_{r,j} \in R_q^{h\times m}$

**if** $\mathcal{Z}[\mathbf{B}_{x,t}] = \bot$ **then**

  $\mathbf{z}' \xleftarrow{\$} R_q^h$

  $\mathcal{Z}[\mathbf{B}_{x,t}] \coloneqq \mathbf{z}'$

$\tilde{\mathbf{e}}' \xleftarrow{\$} \chi_1^h, \tilde{\mathbf{e}} \xleftarrow{\$} \chi^{\ell+m}$

$\mathbf{e}^* \coloneqq \tilde{\mathbf{e}}' - \mathbf{R}_j\tilde{\mathbf{e}} \in R^h$

$\mathbf{u}_x \coloneqq \mathbf{R}_j\mathbf{v}_{k,j} + \mathcal{Z}[\mathbf{B}_{x,t}] + \mathbf{e}^* \in R_q^h$

$\mathsf{stmt}_2 \coloneqq (\mathbf{u}_x, \mathbf{C}_x, \mathbf{c}_k, \mathsf{ck}_2, \mathbf{v}_{k,j}, \mathbf{A}_{r,j})$

$\pi_s \leftarrow \mathsf{NIZK}_2.\mathcal{S}_2(\mathsf{crs}_2, \tau_2, \mathsf{stmt}_2)$

**return** $\mathsf{rep} \coloneqq (\mathbf{u}_x, \pi_s)$

$\mathsf{F.Eval}(\mathsf{sk}, t, x)$

$\mathbf{B}_{x,t} \coloneqq \mathsf{H}(x, t) \in R_q^{h\times m}$

**if** $\mathcal{Z}[\mathbf{B}_{x,t}] = \bot$ **then**

  $\mathbf{z}' \xleftarrow{\$} R_q^h$

  $\mathcal{Z}[\mathbf{B}_{x,t}] \coloneqq \mathbf{z}'$

$\mathbf{z} \coloneqq \lfloor \mathcal{Z}[\mathbf{B}_{x,t}] \rceil_p$

$y \coloneqq \mathsf{RO}_z(t, x, \mathbf{z})$

**return** $y$

Fig. 19: $\mathbf{Hybrid_6}$ of pseudorandomness proof.

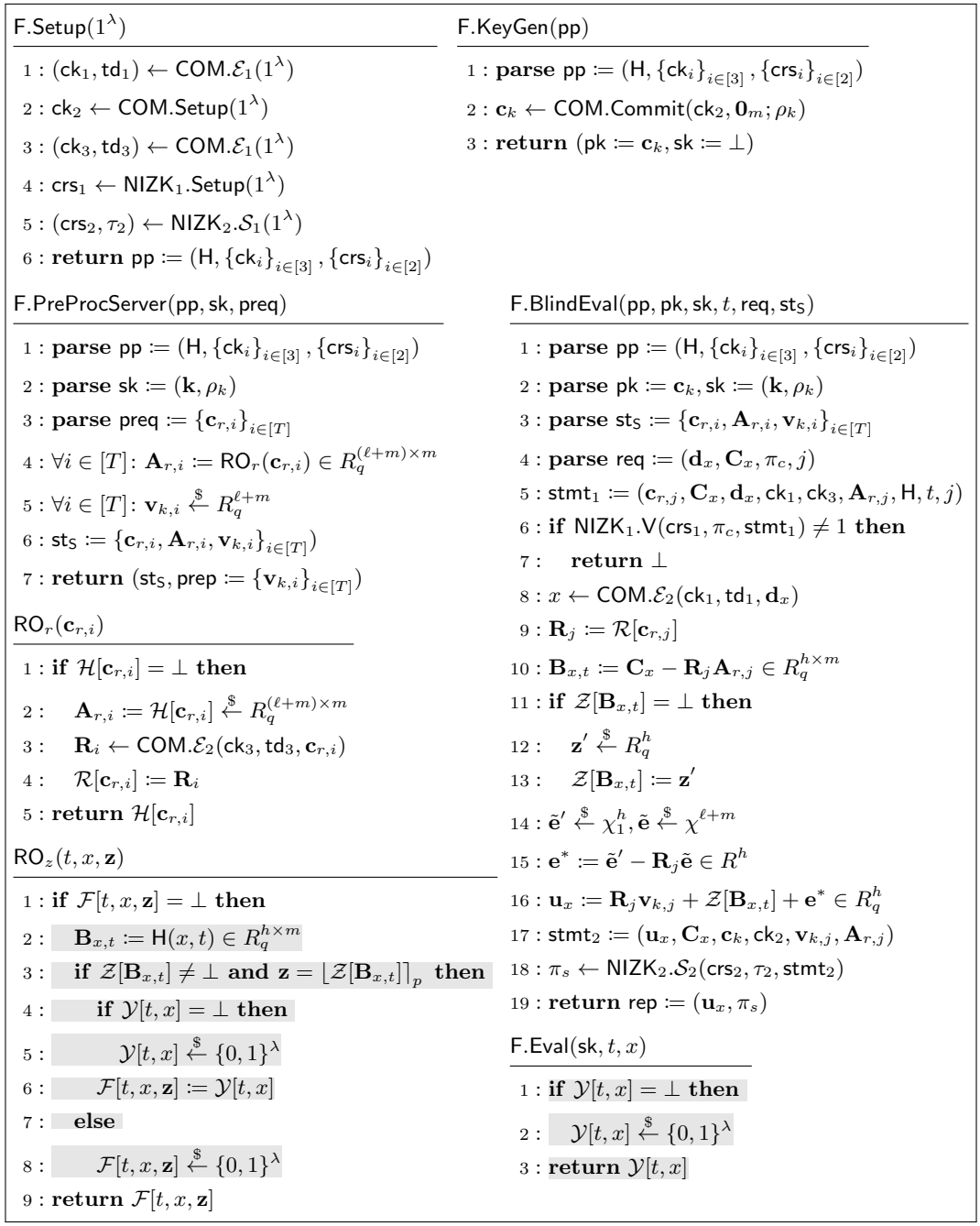


Fig. 20: **Hybrid$_7$** of pseudorandomness proof.

### D.4 Request Privacy Analysis

**Theorem 6.** *The POPRF construction* $\mathsf{F}$ *from Figures 8 and 9 satisfies the request privacy against malicious servers (POPRIV2), given in Definition 4, with random oracles* $\mathsf{RO}_r$ *and* $\mathsf{RO}_z$*, if:*

- *The client argument system* $\mathsf{NIZK}_1$ *is computationally zero-knowledge and the server argument system* $\mathsf{NIZK}_2$ *is computationally sound (Definition 18),*
- *The commitment scheme* $\mathsf{COM}$ *is computationally hiding and extractable (Definitions 15 and 17), and*
- *The* $\mathsf{knMLWE}_{\ell+m,m,h,\chi}$ *assumption holds (Definition 7).*

*More precisely, for any* PPT *adversary* $\mathcal{A}$*, there exist* PPT *adversaries* $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4, \mathcal{B}_5$ *and* $\mathcal{B}_6$ *against the computational zero-knowledge of* $\mathsf{NIZK}_1$*, computational hiding of* $\mathsf{COM}$*, computational soundness of* $\mathsf{NIZK}_2$*, extractability of* $\mathsf{COM}$ *and* $\mathsf{knMLWE}_{m,\ell+m,h,\chi}$ *assumption, respectively, such that*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{po\text{-}priv\text{-}2}}_{\mathsf{F},\mathcal{A},\mathsf{RO}_r,\mathsf{RO}_z}(\lambda) \leq\ & \mathsf{Adv}^{\mathsf{CZK}}_{\mathsf{NIZK}_1,\mathcal{B}_1}(\lambda) + T \cdot Q_{\mathsf{P}} \cdot \mathsf{Adv}^{\mathsf{CH}}_{\mathsf{COM},\mathcal{B}_2}(\lambda) + 2Q_{\mathsf{R}} \cdot \mathsf{Adv}^{\mathsf{CH}}_{\mathsf{COM},\mathcal{B}_3}(\lambda) \\ & + 2Q_{\mathsf{F}} \cdot \mathsf{Adv}^{\mathsf{CS}}_{\mathsf{NIZK}_2,\mathcal{B}_4}(\lambda) + Q^{\mathsf{pk}}_{\mathsf{R}} \cdot \mathsf{Adv}^{\mathsf{Ext}}_{\mathsf{COM},\mathcal{B}_5}(\lambda) + 2Q_{\mathsf{R}} \cdot \mathsf{Adv}^{\mathsf{knMLWE}}_{\mathcal{B}_6}(\lambda)\,,\end{aligned}$$

*where* $Q_{\mathsf{P}}, Q_{\mathsf{R}}$ *and* $Q_{\mathsf{F}}$ *denote the number of* $\mathsf{PreProcClient}, \mathsf{Request}$ *and* $\mathsf{Finalize}$ *oracle queries, respectively, and* $Q^{\mathsf{pk}}_{\mathsf{R}}$ *denotes the number of queries to the* $\mathsf{Request}$ *oracle with distinct* $\mathsf{pk}$ *inputs that the adversary* $\mathcal{A}$ *makes.*

*Proof.* We consider the POPRF request privacy against malicious servers game given in Definition 4. We note that the adversary $\mathcal{A}$ has access to the oracles $\mathsf{PreProcClient}, \mathsf{Request}, \mathsf{Finalize}$ and the random oracles $\mathsf{RO}_r$ and $\mathsf{RO}_z$. The security proof follows in a series of hybrids, where we start with the request privacy against malicious servers game $\mathsf{POPRIV2}^b_{\mathsf{F},\mathcal{A},\mathsf{RO}_r,\mathsf{RO}_z}$, for a random challenge bit $b \in \{0,1\}$, and we gradually apply changes until we end up with a game where the transcript observed by $\mathcal{A}$ is independent of the challenge bit $b$. Let $\mathsf{Adv}_{\mathcal{A},i}(\lambda)$ denote the advantage of $\mathcal{A}$ in $\mathbf{Hybrid_i}$. We consider the hybrids listed below, and for each hybrid we depict the changes using figures, and since the adversary adversarially controlled algorithms are not changed, we do not include them in the figures.

$\mathbf{Hybrid_0}$: This corresponds to the request privacy against malicious servers game $\mathsf{POPRIV2}^b_{\mathsf{F},\mathcal{A},\mathsf{RO}_r,\mathsf{RO}_z}$ (Definition 4), for a random challenge bit $b \in \{0,1\}$, and the interaction follows as in Figures 8 and 9. Hence, it follows that

$$\mathsf{Adv}^{\mathsf{po\text{-}priv\text{-}2}}_{\mathsf{F},\mathcal{A},\mathsf{RO}_r,\mathsf{RO}_z}(\lambda) = \mathsf{Adv}_{\mathcal{A},0}(\lambda)\,.$$

$\mathbf{Hybrid_1}$: In this hybrid, we replace $\mathsf{crs}_1$ and proof $\pi_c$ with a simulated CRS and proof as shown in Figure 21.

In order to indistinguishability between $\mathbf{Hybrid_0}$ and $\mathbf{Hybrid_1}$, we construct a reduction $\mathcal{B}_1$ to the computational zero-knowledge property of $\mathsf{NIZK}_1$. We note that all queries are answered as in $\mathbf{Hybrid_0}$ with the exception of the $\mathsf{Request}$ queries. The only difference here is that $\mathcal{B}_1$ receives the CRS $\mathsf{crs}_1$ from its zero-knowledge challenger instead of computing it via the $\mathsf{NIZK}_1.\mathsf{Setup}$ algorithm. Moreover, when $\mathcal{A}$ makes a $\mathsf{Request}$ query $(i, \mathsf{pk}, \mathsf{prep}, t, x_0, x_1)$, $\mathcal{B}_1$ uses $\mathsf{st}_{\mathsf{C},i}$ (that is the state produced from the $i$-th $\mathsf{PreProcClient}$ oracle call), and proceeds as in $\mathbf{Hybrid_0}$ to compute $(\mathbf{d}_{x_0}, \mathbf{C}_{x_0})$ and $(\mathbf{d}_{x_1}, \mathbf{C}_{x_1})$, but it makes two oracle calls to its zero-knowledge challenger with the inputs $((\mathbf{c}_{r,j}, \mathbf{C}_{x_0}, \mathbf{d}_{x_0}, \mathsf{ck}_1, \mathsf{ck}_3, \mathbf{A}_{r,j}, \mathsf{H}, t, j), (\mathbf{R}_j, x_0, \rho_{r,j}, \rho_{x_0}))$ and $((\mathbf{c}_{r,j+1}, \mathbf{C}_{x_1}, \mathbf{d}_{x_1}, \mathsf{ck}_1, \mathsf{ck}_3, \mathbf{A}_{r,j+1}, \mathsf{H}, t, j+1), (\mathbf{R}_{j+1}, x_1, \rho_{r,j+1}, \rho_{x_1}))$, to obtain the proofs $\pi_{c,0}$ and $\pi_{c,1}$ (without loss of generality we assume $j < T$ here, otherwise $\mathcal{B}_1$ returns $\perp$ and aborts). Finally, $\mathcal{B}_1$ responds to the $\mathsf{Request}$ query with $(\mathsf{req}_b := (\mathbf{d}_{x,b}, \mathbf{C}_{x_b}, \pi_{c,b}, j_b), \mathsf{req}_{1-b} := (\mathbf{d}_{x,1-b}, \mathbf{C}_{x_{1-b}}, \pi_{c,1-b}, j_{1-b}))$.

If the zero-knowledge challenger of $\mathsf{NIZK}_1$ used the honest setup and prover algorithms to generate $\mathsf{crs}_1$ and $(\pi_{c,0}, \pi_{c,1})$, then we are exactly in $\mathbf{Hybrid_0}$, and if it used the simulator, then we are in $\mathbf{Hybrid_1}$. Therefore, if $\mathcal{A}$ can distinguish between the two hybrids with non-negligible advantage, then $\mathcal{B}_1$ can break the zero-knowledge property of $\mathsf{NIZK}_1$. Hence, it follows that

$$\left|\mathsf{Adv}_{\mathcal{A},0}(\lambda) - \mathsf{Adv}_{\mathcal{A},1}(\lambda)\right| \leq \mathsf{Adv}^{\mathsf{CZK}}_{\mathsf{NIZK}_1,\mathcal{B}_1}(\lambda)\,.$$

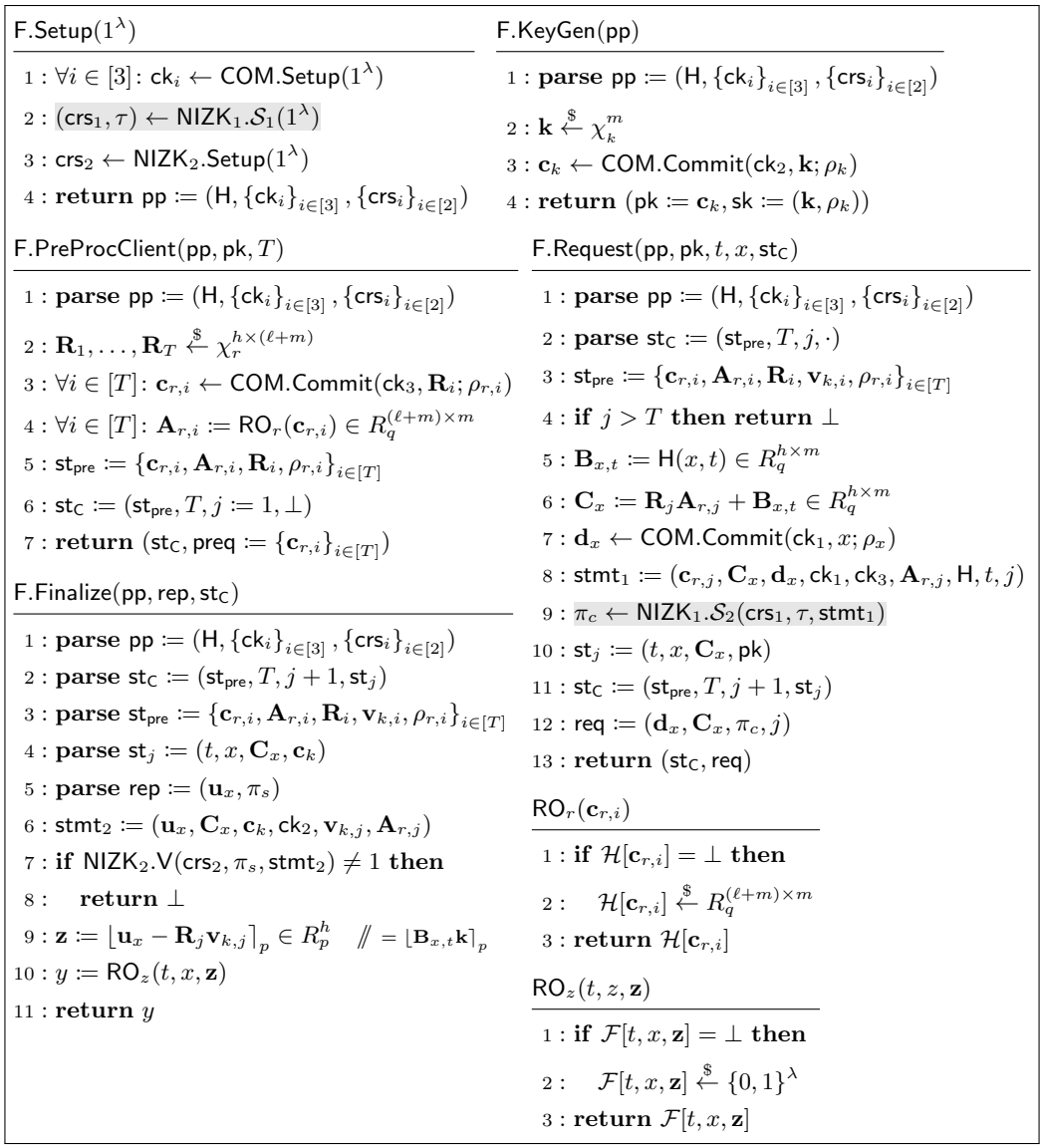


Fig. 21: **Hybrid$_1$** of request privacy proof.

**Hybrid$_2$**: In this hybrid, we commit to $\mathbf{c}_{r,i} \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_3, \mathbf{0}_{h\times(\ell+m)})$, where $\mathbf{0}_{h\times(\ell+m)}$ is an all zero matrix of dimension $h \times (\ell + m)$, instead of computing the commitment $\mathbf{c}_{r,i} \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_3, \mathbf{R}_i)$ as in the previous hybrid. This change is shown in Figure 22.

In order to show indistinguishability between **Hybrid$_1$** and **Hybrid$_2$**, we construct a reduction $\mathcal{B}_2$ to the computational hiding property of COM. We note that the public parameters pp and the responses to all queries are computed as in **Hybrid$_1$**, with the exception of the Request queries. Let $Q_{\mathsf{R}}$ denote the number of PreProcClient queries that the adversary $\mathcal{A}$ makes. Upon receiving a PreProcClient query $(\mathsf{pk}, T)$, $\mathcal{B}_2$ samples $R_i \xleftarrow{\$} \chi_r^{h\times(\ell+m)}$ and submits the pair $(m_{i,0} \coloneqq \mathbf{R}_i, m_{i,1} \coloneqq vec0_{h\times(\ell+m)})$ to the computational hiding challenger of COM, which responds with $\mathbf{c}_i$. At this point $\mathcal{B}_2$ sets $\mathbf{c}_{r,i} \coloneqq \mathbf{c}_i$, and proceeds as in the previous hybrid to simulate the rest of the oracles.

$\mathsf{F.Setup}(1^\lambda)$

$\forall i \in [3]\colon \mathsf{ck}_i \leftarrow \mathsf{COM.Setup}(1^\lambda)$
$(\mathsf{crs}_1, \tau) \leftarrow \mathsf{NIZK}_1.\mathcal{S}_1(1^\lambda)$
$\mathsf{crs}_2 \leftarrow \mathsf{NIZK}_2.\mathsf{Setup}(1^\lambda)$
**return** $\mathsf{pp} \coloneqq (\mathsf{H}, \{\mathsf{ck}_i\}_{i\in[3]}, \{\mathsf{crs}_i\}_{i\in[2]})$

$\mathsf{F.KeyGen}(\mathsf{pp})$

**parse** $\mathsf{pp} \coloneqq (\mathsf{H}, \{\mathsf{ck}_i\}_{i\in[3]}, \{\mathsf{crs}_i\}_{i\in[2]})$
$\mathbf{k} \xleftarrow{\$} \chi_k^m$
$\mathbf{c}_k \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_2, \mathbf{k}; \rho_k)$
**return** $(\mathsf{pk} \coloneqq \mathbf{c}_k, \mathsf{sk} \coloneqq (\mathbf{k}, \rho_k))$

$\mathsf{F.PreProcClient}(\mathsf{pp}, \mathsf{pk}, T)$

**parse** $\mathsf{pp} \coloneqq (\mathsf{H}, \{\mathsf{ck}_i\}_{i\in[3]}, \{\mathsf{crs}_i\}_{i\in[2]})$
$\mathbf{R}_1, \ldots, \mathbf{R}_T \xleftarrow{\$} \chi_r^{h\times(\ell+m)}$
$\forall i \in [T]\colon \mathbf{c}_{r,i} \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_3, \mathbf{0}_{h\times(\ell+m)}; \rho_{r,i})$
$\forall i \in [T]\colon \mathbf{A}_{r,i} \coloneqq \mathsf{RO}_r(\mathbf{c}_{r,i}) \in R_q^{(\ell+m)\times m}$
$\mathsf{st}_{\mathsf{pre}} \coloneqq \{\mathbf{c}_{r,i}, \mathbf{A}_{r,i}, \mathbf{R}_i, \rho_{r,i}\}_{i\in[T]}$
$\mathsf{st}_{\mathsf{C}} \coloneqq (\mathsf{st}_{\mathsf{pre}}, T, j \coloneqq 1, \bot)$
**return** $(\mathsf{st}_{\mathsf{C}}, \mathsf{preq} \coloneqq \{\mathbf{c}_{r,i}\}_{i\in[T]})$

$\mathsf{F.Request}(\mathsf{pp}, \mathsf{pk}, t, x, \mathsf{st}_{\mathsf{C}})$

**parse** $\mathsf{pp} \coloneqq (\mathsf{H}, \{\mathsf{ck}_i\}_{i\in[3]}, \{\mathsf{crs}_i\}_{i\in[2]})$
**parse** $\mathsf{st}_{\mathsf{C}} \coloneqq (\mathsf{st}_{\mathsf{pre}}, T, j, \cdot)$
$\mathsf{st}_{\mathsf{pre}} \coloneqq \{\mathbf{c}_{r,i}, \mathbf{A}_{r,i}, \mathbf{R}_i, \mathbf{v}_{k,i}, \rho_{r,i}\}_{i\in[T]}$
**if** $j > T$ **then return** $\bot$
$\mathbf{B}_{x,t} \coloneqq \mathsf{H}(x,t) \in R_q^{h\times m}$
$\mathbf{C}_x \coloneqq \mathbf{R}_j\mathbf{A}_{r,j} + \mathbf{B}_{x,t} \in R_q^{h\times m}$
$\mathbf{d}_x \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_1, x; \rho_x)$
$\mathsf{stmt}_1 \coloneqq (\mathbf{c}_{r,j}, \mathbf{C}_x, \mathbf{d}_x, \mathsf{ck}_1, \mathsf{ck}_3, \mathbf{A}_{r,j}, \mathsf{H}, t, j)$
$\pi_c \leftarrow \mathsf{NIZK}_1.\mathcal{S}_2(\mathsf{crs}_1, \tau, \mathsf{stmt}_1)$
$\mathsf{st}_j \coloneqq (t, x, \mathbf{C}_x, \mathsf{pk})$
$\mathsf{st}_{\mathsf{C}} \coloneqq (\mathsf{st}_{\mathsf{pre}}, T, j+1, \mathsf{st}_j)$
$\mathsf{req} \coloneqq (\mathbf{d}_x, \mathbf{C}_x, \pi_c, j)$
**return** $(\mathsf{st}_{\mathsf{C}}, \mathsf{req})$

$\mathsf{F.Finalize}(\mathsf{pp}, \mathsf{rep}, \mathsf{st}_{\mathsf{C}})$

**parse** $\mathsf{pp} \coloneqq (\mathsf{H}, \{\mathsf{ck}_i\}_{i\in[3]}, \{\mathsf{crs}_i\}_{i\in[2]})$
**parse** $\mathsf{st}_{\mathsf{C}} \coloneqq (\mathsf{st}_{\mathsf{pre}}, T, j+1, \mathsf{st}_j)$
**parse** $\mathsf{st}_{\mathsf{pre}} \coloneqq \{\mathbf{c}_{r,i}, \mathbf{A}_{r,i}, \mathbf{R}_i, \mathbf{v}_{k,i}, \rho_{r,i}\}_{i\in[T]}$
**parse** $\mathsf{st}_j \coloneqq (t, x, \mathbf{C}_x, \mathbf{c}_k)$
**parse** $\mathsf{rep} \coloneqq (\mathbf{u}_x, \pi_s)$
$\mathsf{stmt}_2 \coloneqq (\mathbf{u}_x, \mathbf{C}_x, \mathbf{c}_k, \mathsf{ck}_2, \mathbf{v}_{k,j}, \mathbf{A}_{r,j})$
**if** $\mathsf{NIZK}_2.\mathsf{V}(\mathsf{crs}_2, \pi_s, \mathsf{stmt}_2) \neq 1$ **then**
  **return** $\bot$
$\mathbf{z} \coloneqq \lfloor \mathbf{u}_x - \mathbf{R}_j\mathbf{v}_{k,j} \rceil_p \in R_p^h \quad /\!\!/ = \lfloor \mathbf{B}_{x,t}\mathbf{k} \rceil_p$
$y \coloneqq \mathsf{RO}_z(t, x, \mathbf{z})$
**return** $y$

$\mathsf{RO}_r(\mathbf{c}_{r,i})$

**if** $\mathcal{H}[\mathbf{c}_{r,i}] = \bot$ **then**
  $\mathcal{H}[\mathbf{c}_{r,i}] \xleftarrow{\$} R_q^{(\ell+m)\times m}$
**return** $\mathcal{H}[\mathbf{c}_{r,i}]$

$\mathsf{RO}_z(t, z, \mathbf{z})$

**if** $\mathcal{F}[t, x, \mathbf{z}] = \bot$ **then**
  $\mathcal{F}[t, x, \mathbf{z}] \xleftarrow{\$} \{0,1\}^\lambda$
**return** $\mathcal{F}[t, x, \mathbf{z}]$

Fig. 22: **Hybrid$_2$** of request privacy proof.

If the computational hiding challenger of COM responds with a commitment to $m_{i,0}$, then we are exactly in **Hybrid$_1$**, and if it responds with a commitment to $m_{i,1}$, then we are in **Hybrid$_2$**. Therefore, if $\mathcal{A}$ can distinguish between the two hybrids with non-negligible advantage, then $\mathcal{B}_2$ can break the computational hiding of COM. Since we consider here a multi-challenge variant of the computational hiding property (concretely, the $T$-challenge variant, where without of loss generality we assume the adversary invokes the PreProcClient oracle with the same value $T$) and the adversary $\mathcal{A}$ makes at most $Q_{\mathsf{P}}$ queries to the PreProcClient oracle, it follows by a standard argument that

$$\left|\mathsf{Adv}_{\mathcal{A},1}(\lambda) - \mathsf{Adv}_{\mathcal{A},2}(\lambda)\right| \leq T \cdot Q_{\mathsf{P}} \cdot \mathsf{Adv}^{\mathsf{CH}}_{\mathsf{COM},\mathcal{B}_2}(\lambda)\,.$$

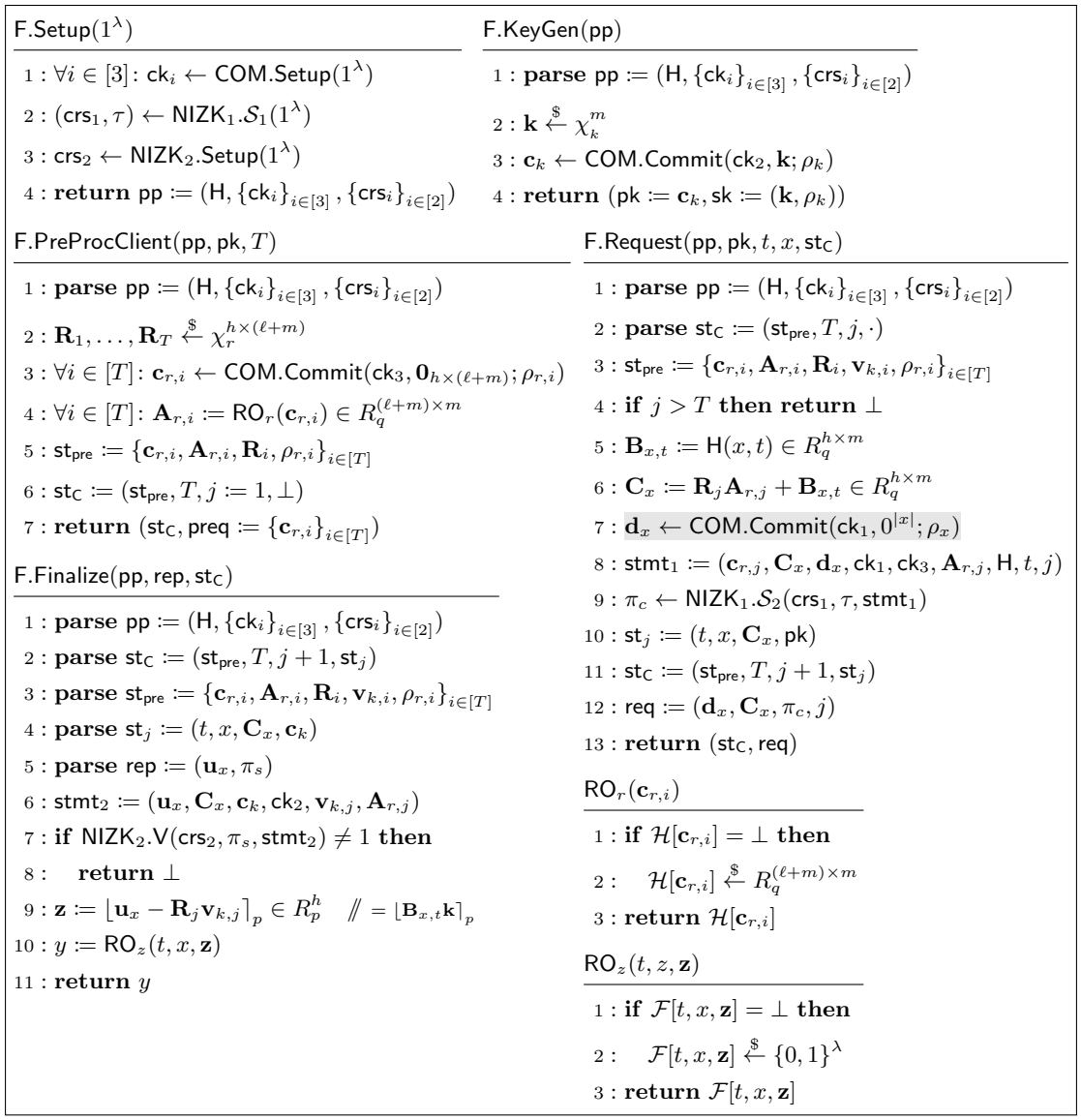


Fig. 23: **Hybrid$_3$** of request privacy proof.

**Hybrid$_3$**: In this hybrid, we commit to $\mathbf{d}_x \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_1, 0^{|x|})$, where $0^{|x|}$ is an all zero bitstring of length $|x|$, instead of computing the commitment $\mathbf{d}_x \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_1, x)$ as in the previous hybrid. This change is shown in Figure 23.

The indistinguishability between **Hybrid$_2$** and **Hybrid$_3$** follow by the computational hiding of COM. Using a similar argumentation as in the previous hybrid, we get that

$$\left|\mathsf{Adv}_{\mathcal{A},2}(\lambda) - \mathsf{Adv}_{\mathcal{A},3}(\lambda)\right| \leq 2Q_{\mathsf{R}} \cdot \mathsf{Adv}^{\mathsf{CH}}_{\mathsf{COM},\mathcal{B}_3}(\lambda),$$

where $\mathcal{B}_3$ is the reduction to the computational hiding of COM, $Q_{\mathsf{R}}$ denotes the number of Request oracle queries that $\mathcal{A}$ makes, and 2 comes from the fact that here we consider a 2-challenge variant of the computational hiding property (since inside the Request oracle we compute two separate requests).

**Hybrid$_4$**: Let BAD be the event that for a Finalize query $(i, j, \mathsf{rep}_0 := (\mathbf{u}_{x,0}, \pi_{s,0}), \mathsf{rep}_1 := (\mathbf{u}_{x,1}, \pi_{s,1}))$, it holds that either

- $\mathsf{NIZK}_2.\mathsf{V}(\mathsf{crs}_2, \pi_{s,0}, (\mathbf{u}_{x,0}, \mathbf{C}_{x,i,j,b}, \mathbf{c}_k, \mathsf{ck}_2, \mathbf{v}_{k,i,j,b}, \mathbf{A}_{r,i,j,b})) = 1$, but $\nexists(\mathbf{k}, \mathbf{e}_s, \mathbf{e}'_s, \rho_k)$ such that $(\mathbf{c}_k = \mathsf{COM.Commit}(\mathsf{ck}_2, \mathbf{k}; \rho_k) \ \wedge \ \mathbf{v}_{k,i,j,b} = \mathbf{A}_{r,i,j,b}\mathbf{k} + \mathbf{e}_s \ \wedge \ \mathbf{u}_{x,0} = \mathbf{C}_{x,i,j,b}\mathbf{k} + \mathbf{e}'_s)$; or
- $\mathsf{NIZK}_2.\mathsf{V}(\mathsf{crs}_2, \pi_{s,1}, (\mathbf{u}_{x,1}, \mathbf{C}_{x,i,j,1-b}, \mathbf{c}_k, \mathsf{ck}_2, \mathbf{v}_{k,i,j,1-b}, \mathbf{A}_{r,i,j,1-b})) = 1$, but $\nexists(\mathbf{k}, \mathbf{e}_s, \mathbf{e}'_s, \rho_k)$ such that $(\mathbf{c}_k = \mathsf{COM.Commit}(\mathsf{ck}_2, \mathbf{k}; \rho_k) \ \wedge \ \mathbf{v}_{k,i,j,1-b} = \mathbf{A}_{r,i,j,1-b}\mathbf{k} + \mathbf{e}_s \ \wedge \ \mathbf{u}_{x,1} = \mathbf{C}_{x,i,j,1-b}\mathbf{k} + \mathbf{e}'_s)$,

where $\mathbf{C}_{x,i,j}$, $\mathbf{A}_{r,i,j}$ and $\mathbf{v}_{k,i,j}$ are values from the $j$-th query to the Request oracle with input $i$ (i.e., for the $i$-th PreProcClient session).

We will prove that the BAD event can only happen with negligible probability due to the soundness of $\mathsf{NIZK}_2$ argument system. Looking ahead, we need here such a reduction to the soundness of $\mathsf{NIZK}_2$ because later in **Hybrid$_5$** we do a reduction to the extractability of the commitment scheme, and there we need to ensure that the adversarial inputs, specifically the public key $\mathsf{pk} := \mathbf{c}_k$ is well-formed.

Clearly, we have that **Hybrid$_3$** and **Hybrid$_4$** are identical until the event BAD happens. Hence, we show that we can bound the probability $\Pr[\mathsf{BAD}]$ by constructing a reduction $\mathcal{B}_4$ to the computational soundness of $\mathsf{NIZK}_2$ argument system. We note that all queries are answered as in **Hybrid$_3$**. Let $Q_\mathsf{F}$ denote the number of Finalize queries that $\mathcal{A}$ performs. $\mathcal{B}_4$ randomly chooses an index $q^* \in [2Q_\mathsf{F}]$ to serve as its guess for the the proof $\pi_s$ that will be forged by the adversary $\mathcal{A}$ (note that each Finalize query includes two proofs, hence the guess from $[2Q_\mathsf{F}]$). Let $(i, j, \mathsf{rep}_0 := (\mathbf{u}_{x,0}, \pi_{s,0}), \mathsf{rep}_1 := (\mathbf{u}_{x,1}, \pi_{s,1}))$ be the $k$-th query to Finalize, where $k = \lceil q^*/2 \rceil$, then $\mathcal{B}_3$ outputs $\pi_{s,0}$ if $q^*$ is odd and outputs $\pi_{s,1}$ otherwise. We emphasize that $\mathcal{B}_3$ avoids recovering any part of the witness $(\mathbf{k}, \mathbf{e}_s, \mathbf{e}'_s, \rho_k)$ (which are computationally infeasible to compute), and instead just guesses which proof would have caused the event BAD to happen. Hence, we have that

$$\left|\mathsf{Adv}_{\mathcal{A},3}(\lambda) - \mathsf{Adv}_{\mathcal{A},4}(\lambda)\right| \leq 2Q_\mathsf{F} \cdot \mathsf{Adv}^{\mathsf{CS}}_{\mathsf{NIZK}_2,\mathcal{B}_4}(\lambda).$$

**Hybrid$_5$**: In this hybrid, we replace the commitment key $\mathsf{ck}_2$ with a trapdoor variant $(\mathsf{ck}_2, \mathsf{td}) \leftarrow \mathsf{COM}.\mathcal{E}_1(1^\lambda)$, and use the trapdoor td to extract the secret key $\mathbf{k}$ from the public key $\mathsf{pk} := \mathbf{c}_k$, i.e., run $\mathbf{k} \leftarrow \mathsf{COM}.\mathcal{E}_2(\mathsf{ck}_2, \mathsf{td}, \mathbf{c}_k)$, and save $\mathbf{k}$ for later use. These changes are depicted in Figure 24.

In order to prove indistinguishability between **Hybrid$_4$** and **Hybrid$_5$** we construct an adversary $\mathcal{B}_5$ against the extractability property of COM. We note that all queries are answered as in **Hybrid$_4$**, and the only change here is that $\mathcal{B}_4$ samples a trapdoor commitment key $(\mathsf{ck}_2, \mathsf{td}) \leftarrow \mathsf{COM}.\mathcal{E}_1(1^\lambda)$, instead of generating $\mathsf{ck}_2$ via the COM.Setup algorithm. Let $Q^{\mathsf{pk}}_\mathsf{R}$ denote the number of Request queries that the adversary $\mathcal{A}$ makes with distinct pk inputs. For each such query $(i, \mathsf{pk}, t, x_0, x_1)$, upon $\mathcal{B}_5$ receiving a Finalize query $(i, j, \mathsf{rep}_0, \mathsf{rep}_1)$, where it corresponds to one of the $Q^{\mathsf{pk}}_\mathsf{R}$ queries made to the Request oracle, $\mathcal{B}_4$ computes $\mathbf{k} \leftarrow \mathsf{COM}.\mathcal{E}_2(\mathsf{ck}_2, \mathsf{td}, \mathsf{pk} := \mathbf{c}_k)$, and stores it in the table $\mathcal{K}[\mathsf{pk}] := \mathbf{k}$ (for later use). The rest of the interaction is identical to **Hybrid$_4$**.

Due to the soundness of the server proof $\pi_s$ we know that the adversarial inputs, and especially the public keys $\mathsf{pk} := \mathbf{c}_k$, are well-formed. Hence, if $\mathcal{A}$ can distinguish between the two hybrids with non-negligible advantage, then $\mathcal{B}_5$ can break the extractability property of COM. Moreover, since the adversary $\mathcal{A}$ makes at most $Q^{\mathsf{pk}}_\mathsf{R}$ queries to the Request oracle with distinct pk, it follows that

$$\left|\mathsf{Adv}_{\mathcal{A},4}(\lambda) - \mathsf{Adv}_{\mathcal{A},5}(\lambda)\right| \leq Q^{\mathsf{pk}}_\mathsf{R} \cdot \mathsf{Adv}^{\mathsf{Ext}}_{\mathsf{COM},\mathcal{B}_5}(\lambda).$$

**Hybrid$_6$**: In this hybrid, we replace $\mathbf{C}_x := \mathbf{R}\mathbf{A}_r + \mathbf{B}_{x,t} \in R_q^{h \times m}$ with $\mathbf{C}^*_x := \mathbf{U} + \mathbf{B}_{x,t} \in R_q^{h \times m}$ for a uniformly random matrix $\mathbf{U} \xleftarrow{\$} R_q^{h \times m}$, as shown in Figure 25.

In order to prove indistinguishability between **Hybrid$_5$** $\approx$ **Hybrid$_6$** we construct and adversary $\mathcal{B}_6$ against the $\mathsf{knMLWE}_{\ell+m,m,h,\chi}$ assumption (Definition 7). We note that the public parameters pp and responses to all queries are computed as in **Hybrid$_5$** with the exception of the Request queries. For each Request query $(i, \mathsf{pk}, t, x_0, x_1)$, $\mathcal{B}_6$ receives from the $\mathsf{knMLWE}_{\ell+m,m,h,\chi}$ challenger the pairs $(\mathbf{A}_0, \mathbf{U}_0)$ and $(\mathbf{A}_1, \mathbf{U}_1)$, which it uses to compute $\mathbf{C}_{x,0} := \mathbf{U}_0 + \mathsf{H}(x_0, t)$ and $\mathbf{C}_{x,1} := \mathbf{U}_1 + \mathsf{H}(x_1, t)$. Note that $\mathbf{C}_{x,0}$ and $\mathbf{C}_{x,1}$ form part of the request that $\mathcal{B}_6$ returns to $\mathcal{A}$ as output of the Request query. Moreover, $\mathcal{B}_6$ programs

$\mathsf{F.Setup}(1^\lambda)$

$1: \forall i \in \{1,3\} : \mathsf{ck}_i \leftarrow \mathsf{COM.Setup}(1^\lambda)$
$2: (\mathsf{ck}_2, \mathsf{td}) \leftarrow \mathsf{COM}.\mathcal{E}_1(1^\lambda)$
$3: (\mathsf{crs}_1, \tau) \leftarrow \mathsf{NIZK}_1.\mathcal{S}_1(1^\lambda)$
$4: \mathsf{crs}_2 \leftarrow \mathsf{NIZK}_2.\mathsf{Setup}(1^\lambda)$
$5:$ **return** $\mathsf{pp} \coloneqq (\mathsf{H}, \{\mathsf{ck}_i\}_{i\in[3]}, \{\mathsf{crs}_i\}_{i\in[2]})$

$\mathsf{F.KeyGen}(\mathsf{pp})$

$1:$ **parse** $\mathsf{pp} \coloneqq (\mathsf{H}, \{\mathsf{ck}_i\}_{i\in[3]}, \{\mathsf{crs}_i\}_{i\in[2]})$
$2: \mathbf{k} \xleftarrow{\$} \chi_k^m$
$3: \mathbf{c}_k \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_2, \mathbf{k}; \rho_k)$
$4:$ **return** $(\mathsf{pk} \coloneqq \mathbf{c}_k, \mathsf{sk} \coloneqq (\mathbf{k}, \rho_k))$

$\mathsf{F.PreProcClient}(\mathsf{pp}, \mathsf{pk}, T)$

$1:$ **parse** $\mathsf{pp} \coloneqq (\mathsf{H}, \{\mathsf{ck}_i\}_{i\in[3]}, \{\mathsf{crs}_i\}_{i\in[2]})$
$2: \mathbf{R}_1, \ldots, \mathbf{R}_T \xleftarrow{\$} \chi_r^{h\times(\ell+m)}$
$3: \forall i \in [T]: \mathbf{c}_{r,i} \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_3, \mathbf{0}_{h\times(\ell+m)}; \rho_{r,i})$
$4: \forall i \in [T]: \mathbf{A}_{r,i} \coloneqq \mathsf{RO}_r(\mathbf{c}_{r,i}) \in R_q^{(\ell+m)\times m}$
$5: \mathsf{st}_{\mathsf{pre}} \coloneqq \{\mathbf{c}_{r,i}, \mathbf{A}_{r,i}, \mathbf{R}_i, \rho_{r,i}\}_{i\in[T]}$
$6: \mathsf{st}_\mathsf{C} \coloneqq (\mathsf{st}_{\mathsf{pre}}, T, j \coloneqq 1, \bot)$
$7:$ **return** $(\mathsf{st}_\mathsf{C}, \mathsf{preq} \coloneqq \{\mathbf{c}_{r,i}\}_{i\in[T]})$

$\mathsf{F.Finalize}(\mathsf{pp}, \mathsf{rep}, \mathsf{st}_\mathsf{C})$

$1:$ **parse** $\mathsf{pp} \coloneqq (\mathsf{H}, \{\mathsf{ck}_i\}_{i\in[3]}, \{\mathsf{crs}_i\}_{i\in[2]})$
$2:$ **parse** $\mathsf{st}_\mathsf{C} \coloneqq (\mathsf{st}_{\mathsf{pre}}, T, j+1, \mathsf{st}_j)$
$3:$ **parse** $\mathsf{st}_{\mathsf{pre}} \coloneqq \{\mathbf{c}_{r,i}, \mathbf{A}_{r,i}, \mathbf{R}_i, \mathbf{v}_{k,i}, \rho_{r,i}\}_{i\in[T]}$
$4:$ **parse** $\mathsf{st}_j \coloneqq (t, x, \mathbf{C}_x, \mathbf{c}_k)$
$5:$ **parse** $\mathsf{rep} \coloneqq (\mathbf{u}_x, \pi_s)$
$6: \mathsf{stmt}_2 \coloneqq (\mathbf{u}_x, \mathbf{C}_x, \mathbf{c}_k, \mathsf{ck}_2, \mathbf{v}_{k,j}, \mathbf{A}_{r,j})$
$7:$ **if** $\mathsf{NIZK}_2.\mathsf{V}(\mathsf{crs}_2, \pi_s, \mathsf{stmt}_2) \neq 1$ **then**
$8:$ **return** $\bot$
$9:$ **if** $\mathcal{K}[\mathsf{pk}] = \bot$ **then**
$10:$ $\mathbf{k} \leftarrow \mathsf{COM}.\mathcal{E}_2(\mathsf{ck}_2, \mathsf{td}, \mathbf{c}_k)$
$11:$ $\mathcal{K}[\mathsf{pk}] \coloneqq \mathbf{k}$
$12: \mathbf{z} \coloneqq \lfloor \mathbf{u}_x - \mathbf{R}_j \mathbf{v}_{k,j} \rceil_p \in R_p^h \quad /\!/ = \lfloor \mathbf{B}_{x,t}\mathbf{k} \rceil_p$
$13: y \coloneqq \mathsf{RO}_z(t, x, \mathbf{z})$
$14:$ **return** $y$

$\mathsf{F.Request}(\mathsf{pp}, \mathsf{pk}, t, x, \mathsf{st}_\mathsf{C})$

$1:$ **parse** $\mathsf{pp} \coloneqq (\mathsf{H}, \{\mathsf{ck}_i\}_{i\in[3]}, \{\mathsf{crs}_i\}_{i\in[2]})$
$2:$ **parse** $\mathsf{st}_\mathsf{C} \coloneqq (\mathsf{st}_{\mathsf{pre}}, T, j, \cdot)$
$3: \mathsf{st}_{\mathsf{pre}} \coloneqq \{\mathbf{c}_{r,i}, \mathbf{A}_{r,i}, \mathbf{R}_i, \mathbf{v}_{k,i}, \rho_{r,i}\}_{i\in[T]}$
$4:$ **if** $j > T$ **then return** $\bot$
$5: \mathbf{B}_{x,t} \coloneqq \mathsf{H}(x,t) \in R_q^{h\times m}$
$6: \mathbf{C}_x \coloneqq \mathbf{R}_j \mathbf{A}_{r,j} + \mathbf{B}_{x,t} \in R_q^{h\times m}$
$7: \mathbf{d}_x \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_1, 0^{|x|}; \rho_x)$
$8: \mathsf{stmt}_1 \coloneqq (\mathbf{c}_{r,j}, \mathbf{C}_x, \mathbf{d}_x, \mathsf{ck}_1, \mathsf{ck}_3, \mathbf{A}_{r,j}, \mathsf{H}, t, j)$
$9: \pi_c \leftarrow \mathsf{NIZK}_1.\mathcal{S}_2(\mathsf{crs}_1, \tau, \mathsf{stmt}_1)$
$10: \mathsf{st}_j \coloneqq (t, x, \mathbf{C}_x, \mathsf{pk})$
$11: \mathsf{st}_\mathsf{C} \coloneqq (\mathsf{st}_{\mathsf{pre}}, T, j+1, \mathsf{st}_j)$
$12: \mathsf{req} \coloneqq (\mathbf{d}_x, \mathbf{C}_x, \pi_c, j)$
$13:$ **return** $(\mathsf{st}_\mathsf{C}, \mathsf{req})$

$\mathsf{RO}_r(\mathbf{c}_{r,i})$

$1:$ **if** $\mathcal{H}[\mathbf{c}_{r,i}] = \bot$ **then**
$2:$ $\mathcal{H}[\mathbf{c}_{r,i}] \xleftarrow{\$} R_q^{(\ell+m)\times m}$
$3:$ **return** $\mathcal{H}[\mathbf{c}_{r,i}]$

$\mathsf{RO}_z(t, z, \mathbf{z})$

$1:$ **if** $\mathcal{F}[t, x, \mathbf{z}] = \bot$ **then**
$2:$ $\mathcal{F}[t, x, \mathbf{z}] \xleftarrow{\$} \{0,1\}^\lambda$
$3:$ **return** $\mathcal{F}[t, x, \mathbf{z}]$

Fig. 24: **Hybrid$_5$** of request privacy proof.

the random oracle $\mathsf{RO}_r$, such that on input the pair $\mathbf{c}_{r,i}$ it returns $\mathbf{A}_i$ provided by the $\mathsf{knMLWE}_{\ell+m,m,h,\chi}$ challenger (we note that this change is not highlighted in Figure 25, because the $\mathbf{A}_i$ values returned by the $\mathsf{knMLWE}_{\ell+m,m,h,\chi}$ challenger are also uniformly random values from $R_q^{(\ell+m)\times m}$). Upon receiving a Finalize query $(i, j, \mathsf{rep}_0 \coloneqq (\mathbf{u}_{x,0}, \pi_{s,0}), \mathsf{rep}_1 \coloneqq (\mathbf{u}_{x,0}, \pi_{s,0}))$ from $\mathcal{A}$, $\mathcal{B}_6$ proceeds as in the previous hybrid, to obtain $\mathsf{pk} \coloneqq \mathbf{c}_k$ from the state $\mathsf{st}_j$, and then extracts $\mathbf{k}$ from $\mathsf{pk}$ using the commitment trapdoor. Additionally, $\mathcal{B}_5$ uses the secret key $\mathbf{k}$ to compute the values $\mathbf{z}_0 \coloneqq \lfloor \mathbf{u}_{x,0} - \mathbf{U}_0\mathbf{k} - \mathbf{R}_{i,j,0}(\mathbf{v}_{k,i,j,0} - \mathbf{A}_{r,i,j,0}\mathbf{k}) \rceil_p$ and $\mathbf{z}_1 \coloneqq \lfloor \mathbf{u}_{x,1} - \mathbf{U}_1\mathbf{k} - \mathbf{R}_{i,j,1}(\mathbf{v}_{k,i,j,1} - \mathbf{A}_{r,i,j,1}\mathbf{k}) \rceil_p$, which constitute an alternative way of computing the correct $\mathbf{z}_0$ and $\mathbf{z}_1$ values with the help of the secret key $\mathbf{k}$ and the randomnesses $\mathbf{R}_{i,j,0}$ and $\mathbf{R}_{i,j,1}$ that were

F.Setup($1^\lambda$)

$\forall i \in \{1,3\} : \mathsf{ck}_i \leftarrow \mathsf{COM.Setup}(1^\lambda)$
$(\mathsf{ck}_2, \mathsf{td}) \leftarrow \mathsf{COM}.\mathcal{E}_1(1^\lambda)$
$(\mathsf{crs}_1, \tau) \leftarrow \mathsf{NIZK}_1.\mathcal{S}_1(1^\lambda)$
$\mathsf{crs}_2 \leftarrow \mathsf{NIZK}_2.\mathsf{Setup}(1^\lambda)$
**return** $\mathsf{pp} \coloneqq (\mathsf{H}, \{\mathsf{ck}_i\}_{i\in[3]}, \{\mathsf{crs}_i\}_{i\in[2]})$

F.KeyGen(pp)

**parse** $\mathsf{pp} \coloneqq (\mathsf{H}, \{\mathsf{ck}_i\}_{i\in[3]}, \{\mathsf{crs}_i\}_{i\in[2]})$
$\mathbf{k} \xleftarrow{\$} \chi_k^m$
$\mathbf{c}_k \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_2, \mathbf{k}; \rho_k)$
**return** $(\mathsf{pk} \coloneqq \mathbf{c}_k, \mathsf{sk} \coloneqq (\mathbf{k}, \rho_k))$

F.PreProcClient(pp, pk, $T$)

**parse** $\mathsf{pp} \coloneqq (\mathsf{H}, \{\mathsf{ck}_i\}_{i\in[3]}, \{\mathsf{crs}_i\}_{i\in[2]})$
$\mathbf{R}_1, \ldots, \mathbf{R}_T \xleftarrow{\$} \chi_r^{h\times(\ell+m)}$
$\forall i \in [T]$: $\mathbf{c}_{r,i} \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_3, \mathbf{0}_{h\times(\ell+m)}; \rho_{r,i})$
$\forall i \in [T]$: $\mathbf{A}_{r,i} \coloneqq \mathsf{RO}_r(\mathbf{c}_{r,i}) \in R_q^{(\ell+m)\times m}$
$\mathsf{st}_{\mathsf{pre}} \coloneqq \{\mathbf{c}_{r,i}, \mathbf{A}_{r,i}, \mathbf{R}_i, \rho_{r,i}\}_{i\in[T]}$
$\mathsf{st}_{\mathsf{C}} \coloneqq (\mathsf{st}_{\mathsf{pre}}, T, j \coloneqq 1, \bot)$
**return** $(\mathsf{st}_{\mathsf{C}}, \mathsf{preq} \coloneqq \{\mathbf{c}_{r,i}\}_{i\in[T]})$

F.Finalize(pp, rep, $\mathsf{st}_{\mathsf{C}}$)

**parse** $\mathsf{pp} \coloneqq (\mathsf{H}, \{\mathsf{ck}_i\}_{i\in[3]}, \{\mathsf{crs}_i\}_{i\in[2]})$
**parse** $\mathsf{st}_{\mathsf{C}} \coloneqq (\mathsf{st}_{\mathsf{pre}}, T, j+1, \mathsf{st}_j)$
**parse** $\mathsf{st}_{\mathsf{pre}} \coloneqq \{\mathbf{c}_{r,i}, \mathbf{A}_{r,i}, \mathbf{R}_i, \mathbf{v}_{k,i}, \rho_{r,i}\}_{i\in[T]}$
**parse** $\mathsf{st}_j \coloneqq (t, x, \mathbf{C}_x, \mathbf{c}_k)$
**parse** $\mathsf{rep} \coloneqq (\mathbf{u}_x, \pi_s)$
$\mathsf{stmt}_2 \coloneqq (\mathbf{u}_x, \mathbf{C}_x, \mathbf{c}_k, \mathsf{ck}_2, \mathbf{v}_{k,j}, \mathbf{A}_{r,j})$
**if** $\mathsf{NIZK}_2.\mathsf{V}(\mathsf{crs}_2, \pi_s, \mathsf{stmt}_2) \neq 1$ **then**
  **return** $\bot$
**if** $\mathcal{K}[\mathsf{pk}] = \bot$ **then**
  $\mathbf{k} \leftarrow \mathsf{COM}.\mathcal{E}_2(\mathsf{ck}_2, \mathsf{td}, \mathbf{c}_k)$
  $\mathcal{K}[\mathsf{pk}] \coloneqq \mathbf{k}$
$\mathbf{z} \coloneqq \lfloor \mathbf{u}_x - \mathbf{U}\mathcal{K}[\mathsf{pk}] - \mathbf{R}_j(\mathbf{v}_{k,j} - \mathbf{A}_{r,j}\mathbf{k}) \rceil_p \in R_p^h$
$y \coloneqq \mathsf{RO}_z(t, x, \mathbf{z})$
**return** $y$

F.Request(pp, pk, $t, x, \mathsf{st}_{\mathsf{C}}$)

**parse** $\mathsf{pp} \coloneqq (\mathsf{H}, \{\mathsf{ck}_i\}_{i\in[3]}, \{\mathsf{crs}_i\}_{i\in[2]})$
**parse** $\mathsf{st}_{\mathsf{C}} \coloneqq (\mathsf{st}_{\mathsf{pre}}, T, j, \cdot)$
$\mathsf{st}_{\mathsf{pre}} \coloneqq \{\mathbf{c}_{r,i}, \mathbf{A}_{r,i}, \mathbf{R}_i, \mathbf{v}_{k,i}, \rho_{r,i}\}_{i\in[T]}$
**if** $j > T$ **then return** $\bot$
$\mathbf{B}_{x,t} \coloneqq \mathsf{H}(x,t) \in R_q^{h\times m}$
$\mathbf{U} \xleftarrow{\$} R_q^{h\times m}$
$\mathbf{C}_x \coloneqq \mathbf{U} + \mathbf{B}_{x,t} \in R_q^{h\times m}$
$\mathbf{d}_x \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_1, 0^{|x|}; \rho_x)$
$\mathsf{stmt}_1 \coloneqq (\mathbf{c}_{r,j}, \mathbf{C}_x, \mathbf{d}_x, \mathsf{ck}_1, \mathsf{ck}_3, \mathbf{A}_{r,j}, \mathsf{H}, t, j)$
$\pi_c \leftarrow \mathsf{NIZK}_1.\mathcal{S}_2(\mathsf{crs}_1, \tau, \mathsf{stmt}_1)$
$\mathsf{st}_j \coloneqq (t, x, \mathbf{C}_x, \mathsf{pk})$
$\mathsf{st}_{\mathsf{C}} \coloneqq (\mathsf{st}_{\mathsf{pre}}, T, j+1, \mathsf{st}_j)$
$\mathsf{req} \coloneqq (\mathbf{d}_x, \mathbf{C}_x, \pi_c, j)$
**return** $(\mathsf{st}_{\mathsf{C}}, \mathsf{req})$

$\mathsf{RO}_r(\mathbf{c}_{r,i})$

**if** $\mathcal{H}[\mathbf{c}_{r,i}] = \bot$ **then**
  $\mathcal{H}[\mathbf{c}_{r,i}] \xleftarrow{\$} R_q^{(\ell+m)\times m}$
**return** $\mathcal{H}[\mathbf{c}_{r,i}]$

$\mathsf{RO}_z(t, z, \mathbf{z})$

**if** $\mathcal{F}[t, x, \mathbf{z}] = \bot$ **then**
  $\mathcal{F}[t, x, \mathbf{z}] \xleftarrow{\$} \{0,1\}^\lambda$
**return** $\mathcal{F}[t, x, \mathbf{z}]$

Fig. 25: $\mathbf{Hybrid_6}$ of request privacy proof.

sampled during the Request query. Finally, $\mathcal{B}$ computes the output $y_0$ and $y_1$ as before and returns it to the adversary $\mathcal{A}$.

If the $\mathsf{knMLWE}_{\ell+m,m,h,\chi}$ challenger provided a knapsack MLWE instance, i.e., $\mathbf{U} \coloneqq \mathbf{RA}$, for some $\mathbf{R} \in R_q^{h\times(\ell+m)}$, then the view is as in $\mathbf{Hybrid_5}$, and if it provided a uniformly random matrix $\mathbf{U} \in R_q^{h\times m}$, then the view is as in $\mathbf{Hybrid_6}$. Therefore, if $\mathcal{A}$ can distinguish between the two hybrids with non-negligible advantage, then $\mathcal{B}_6$ can break the $\mathsf{knMLWE}_{\ell+m,m,h,\chi}$ assumption. Since we consider here a multi-challenge variant of the $\mathsf{knMLWE}_{\ell+m,m,h,\chi}$ problem (concretely, the 2-challenge variant) and the adversary $\mathcal{A}$ makes

at most $Q_{\mathsf{R}}$ queries to the $\mathsf{Request}$ oracle, it follows by a standard argument that

$$\left|\mathsf{Adv}_{\mathcal{A},5}(\lambda) - \mathsf{Adv}_{\mathcal{A},6}(\lambda)\right| \leq 2Q_{\mathsf{R}} \cdot \mathsf{Adv}^{\mathsf{knMLWE}_{\ell+m,m,h,\chi}}_{\mathcal{B}_6}(\lambda).$$

We note that at this point the transcript observed by the adversary $\mathcal{A}$ is independent of the challenge bit $b$, and hence $\mathsf{Adv}_{\mathcal{A},6}(\lambda) = 1/2$. Putting everything together, we obtain

$$\begin{aligned}\mathsf{Adv}^{\mathsf{po\text{-}priv\text{-}2}}_{\mathsf{F},\mathcal{A},\mathsf{RO}_r,\mathsf{RO}_z}(\lambda) \leq\ & \mathsf{Adv}^{\mathsf{CZK}}_{\mathsf{NIZK}_1,\mathcal{B}_1}(\lambda) + T \cdot Q_{\mathsf{P}} \cdot \mathsf{Adv}^{\mathsf{CH}}_{\mathsf{COM},\mathcal{B}_2}(\lambda) + 2Q_{\mathsf{R}} \cdot \mathsf{Adv}^{\mathsf{CH}}_{\mathsf{COM},\mathcal{B}_3}(\lambda) \\ & + 2Q_{\mathsf{F}} \cdot \mathsf{Adv}^{\mathsf{CS}}_{\mathsf{NIZK}_2,\mathcal{B}_4}(\lambda) + Q^{\mathsf{pk}}_{\mathsf{R}} \cdot \mathsf{Adv}^{\mathsf{Ext}}_{\mathsf{COM},\mathcal{B}_5}(\lambda) + 2Q_{\mathsf{R}} \cdot \mathsf{Adv}^{\mathsf{knMLWE}}_{\mathcal{B}_6}(\lambda),\end{aligned}$$

as claimed. This completes the proof of Theorem 6. □

# E Issues in Prior Constructions and Models

## E.1 Issues in [ADDS21] and [ADDG24]

We describe here two flaws present in the security proofs of the existing lattice-based OPRF constructions. The first flaw is from the [ADDS21] paper, which aims to prove their construction secure in the simulation based setting (we refer the reader to [ADDS21] for details regarding the security model). The flaw appears during the Query phase of the malicious client security proof (given in [ADDS21, Lemma 8]), where for each query $(\mathbf{c}_x, \pi)$ from the malicious client, the simulator needs to extract the client's secret input $x$ from the NIZK argument $\pi$ in order to complete the reduction. However, the authors make use of a NIZK argument system that is based on Fiat-Shamir transform [YAZ+19], and hence, is not straight-line extractable, i.e., cannot extract the witness without rewinding. This means that each query of the malicious client incurs a loss in the soundness of the NIZK argument system, and since a malicious client can make $Q = \mathsf{poly}(\lambda)$ distinct queries, it means we will end up with an exponential decay in soundness. We note that this issue can be fixed by using the Katsumata transform [Kat21] to obtain a lattice-based straight-line extractable NIZK argument system, albeit by incurring a performance penalty.

The second flaw appears in the recent [ADDG24] paper. Specifically, during the security proof of request privacy against malicious servers (POPRIV2) (given in [ADDG24, Theorem 4]), one of the reductions requires access to the server's secret key $\mathsf{sk}$. The authors attempt to solve this dilemma by requiring the server to give its $\mathsf{sk}$ to the reduction algorithm, however, this is in stark contrast to the request privacy game given in Definition 4 (also used in [ADDG24]). Since their construction already includes NIZK proofs, one potential way to patch this issue is to instead extract the $\mathsf{sk}$ from a NIZK proof. However, similar to the previously explained issue, one needs to consider here a straight-line extractable proof system, which degrades the performance of the scheme. An alternative solution is to use an extractable commitment scheme and include a commitment to $\mathsf{sk}$ as part of the public key $\mathsf{pk}$, which is exactly what we do in our construction given in Section 2.1, and during our request privacy security proof given in Appendix D.4.

## E.2 Uniqueness and Key Binding

In [TCR+22], the authors defined a property called *uniqueness*, which ensures that POPRF outputs are unique even in the case of a malicious server.

**Definition 20 (Uniqueness [TCR+22]).** *We say that a partial oblivious PRF (POPRF)* $\mathsf{F}$ *is unique against malicious servers, if for all* PPT *adversaries* $\mathcal{A}$ *the following advantage is* $\mathsf{negl}(\lambda)$*,*

$$\mathsf{Adv}^{\mathsf{po\text{-}uniq}}_{\mathsf{F},\mathcal{A},\mathsf{RO}}(\lambda) = \Pr\left[\mathsf{POUNIQ}_{\mathsf{F},\mathcal{A},\mathsf{RO}}(\lambda) = 1\right],$$

*where the experiment* $\mathsf{POUNIQ}$ *is defined as follows:*

$\mathsf{POUNIQ}_{\mathsf{F},\mathcal{A},\mathsf{RO}}(\lambda)$

1 : $R := [\cdot], q := 0$
2 : $\mathsf{pp} \leftarrow \mathsf{F.Setup}(1^\lambda)$
3 : $\mathsf{st}_{\mathsf{RO}} \leftarrow \mathsf{RO.Init}(1^\lambda)$
4 : $(i, j, \mathsf{rep}_i, \mathsf{rep}_j) \leftarrow \mathcal{A}^{\mathsf{Request},\mathsf{RO}}(\mathsf{pp})$
5 : **assert** $1 \leq i \leq j \leq q$
6 : $(\mathsf{st}_i, \mathsf{pk}_i, t_i, x_i) := R[i]$
7 : $(\mathsf{st}_j, \mathsf{pk}_j, t_j, x_j) := R[j]$
8 : $y_i \leftarrow \mathsf{F.Finalize}^{\mathsf{RO}}_{\mathsf{pp}}(\mathsf{rep}_i, \mathsf{st}_i)$
9 : $y_j \leftarrow \mathsf{F.Finalize}^{\mathsf{RO}}_{\mathsf{pp}}(\mathsf{rep}_j, \mathsf{st}_j)$
10 : **if** $y_i = \bot \vee y_j = \bot$ **then**
11 : **return** 0
12 : **return** $((\mathsf{pk}_i, t_i, x_i) = (\mathsf{pk}_j, t_j, x_j)) \wedge (y_i \neq y_j)$

$\mathsf{Request}(\mathsf{sk}, \mathsf{aux}, \mathsf{pk}, t, x)$

1 : **assert** $\mathsf{F.Wellformed}(\mathsf{sk}, \mathsf{pk}, \mathsf{aux})$
2 : $(\mathsf{st}, \mathsf{req}) \leftarrow \mathsf{F.Request}^{\mathsf{RO}}_{\mathsf{pp}}(\mathsf{pk}, t, x)$
3 : $q := q + 1$
4 : $R[q] := (\mathsf{st}, \mathsf{pk}, t, x)$
5 : **return** req

*The highlighted part is included to address rogue key attacks using the knowledge of secret key model [Bol03]. If the* Wellformed *predicate does not require any auxiliary information to verify the secret/public key pair* $(\mathsf{sk}, \mathsf{pk})$*, then we consider that* $\mathsf{aux} := \bot$*.*

Tyagi et al. [TCR+22] showed that uniqueness follows from the correctness and request privacy against malicious servers (POPRIV2) of the POPRF. However, since correctness only holds against well-formed public keys and uniqueness definition considers a malicious adversary, they proved this in the knowledge of secret key (KOSK) model [Bol03]. In KOSK, the adversary reveals its secret key and this is used to check the well-formedness of the secret/public key pair (e.g., by using the Wellformed predicate in Definition 20). We note that such a model can be instantiated by including an extractable proof of knowledge or commitment, from which the secret key can be extracted and the proof may proceed as in the KOSK model.

Nevertheless, the reduction from uniqueness to request privacy that was given by Tyagi et al. [TCR+22, Appendix H], only works when there is a bijection between the secret and public keys, i.e., for each public key pk there is a unique sk. This comes from the fact that during the reduction they consider that the output of a valid uniqueness adversary would result in two tuples $(\mathsf{pk}_i, \mathsf{sk}_i, t_i, x_i) = (\mathsf{pk}_j, \mathsf{sk}_j, t_j, x_j)$. However, the uniqueness definition (see Definition 20) only guarantees that $(\mathsf{pk}_i, t_i, x_i) = (\mathsf{pk}_j, t_j, x_j)$, i.e., we do not have the guarantee that $\mathsf{sk}_i = \mathsf{sk}_j$. Although, this holds in group-based OPRF constructions, such as in [TCR+22], this does not necessarily hold in lattice-based constructions. For example, in [ADDS21] the public key is of the form $\mathbf{c} := \mathbf{a} \cdot k + \mathbf{e} \bmod q$, such that for a fixed (non-zero) $\mathbf{a}$, there may exist many pairs of secret keys $(k, \mathbf{e})$ that satisfy the public key equation.

In order to provide a more robust uniqueness guarantees, we define here an additional property for a POPRF called *key binding*, which ensures that a public key is bound to a particular secret key.

**Definition 21 (Key Binding).** *We say that a partial oblivious PRF (POPRF)* F *is key binding with respect to a* Wellformed *predicate, if for all* PPT *adversaries* $\mathcal{A}$ *the following advantage is* $\mathsf{negl}(\lambda)$*,*

$$\mathsf{Adv}^{\mathsf{po\text{-}key\text{-}bind}}_{\mathsf{F},\mathcal{A},\mathsf{RO}}(\lambda) = \Pr\left[\mathsf{POKBIND}_{\mathsf{F},\mathcal{A},\mathsf{RO}}(\lambda) = 1\right],$$

*where the experiment* POKBIND *is defined as follows:*

$\mathsf{POKBIND}_{\mathsf{F},\mathcal{A},\mathsf{RO}}(\lambda)$

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

$\mathsf{st}_{\mathsf{RO}} \leftarrow \mathsf{RO.Init}(1^\lambda)$

$(\mathsf{pk}, \mathsf{sk}, \mathsf{sk}', \mathsf{aux}, \mathsf{aux}') \leftarrow \mathcal{A}^{\mathsf{RO}}(\mathsf{pp})$

**assert** $\mathsf{sk} \neq \mathsf{sk}'$

**return** $\mathsf{F.Wellformed}(\mathsf{sk}, \mathsf{pk}, \mathsf{aux}) \wedge \mathsf{F.Wellformed}(\mathsf{sk}', \mathsf{pk}, \mathsf{aux}')$

Next, we show that our construction from Figure 9 satisfies key binding.

**Theorem 7.** *If* COM *is computationally binding, then the OPRF construction* F *from Figure 9 is key binding.*

*Proof.* First we define a Wellformed predicate for our key binding definition. In our construction the secret key is of the form $\mathsf{sk} := \mathbf{k}$, for $\mathbf{k} \xleftarrow{\$} \chi_k^m$, and the public key is computed as $\mathsf{pk} := \mathbf{c}_k \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_2, \mathbf{k})$, for some commitment key $\mathsf{ck}_2 \leftarrow \mathsf{COM.Setup}(1^\lambda)$. Hence, our Wellformed predicate can be defined as the verification algorithm of the commitment scheme COM, i.e., $\mathsf{F.Wellformed}(\mathsf{sk}, \mathsf{pk}, \mathsf{aux}) = \mathsf{COM.Verify}(\mathsf{ck}_2, \mathbf{c}_k, \rho_k, \mathbf{k})$, where $\mathsf{aux} := \rho_k$ is the decommitment information.

In order to show that our construction satisfies key binding we construct a reduction $\mathcal{B}$ to the computational binding property of COM. $\mathcal{B}$ receives the commitment key $\mathsf{ck}^*$ from the computational binding challenger of COM, sets $\mathsf{ck}_2 := \mathsf{ck}^*$, generates the rest of the public parameters pp as in Figure 9, and sends pp to $\mathcal{A}$. Upon receiving the tuple $(\mathsf{pk} := \mathbf{c}_k, \mathsf{sk} := \mathbf{k}, \mathsf{sk}' := \mathbf{k}', \mathsf{aux} := \rho_k, \mathsf{aux} := \rho'_k)$ from $\mathcal{A}$, $\mathcal{B}$ forwards the tuple $(\mathbf{c}_k, \rho_k, \rho'_k, \mathbf{k}, \mathbf{k}')$ to the computational binding challenger of COM.

We note that the winning conditions for the key binding of F (Definition 21) and computational binding of COM (Definition 16) are identical in this case. Hence, if $\mathcal{A}$ can win the key binding experiment with non-negligible advantage, then $\mathcal{B}$ can win the computational binding experiment with non-negligible advantage. □

**Theorem 8.** *For every* PPT *adversary* $\mathcal{A}$ *against the uniqueness of* F*, there exist* PPT *adversaries* $\mathcal{B}_1$ *and* $\mathcal{B}_2$ *against the key binding and request privacy of* F*, respectively, such that*

$$\mathsf{Adv}^{\text{po-uniq}}_{\mathsf{F},\mathcal{A},\mathsf{RO}}(\lambda) \leq \mathsf{Adv}^{\text{po-key-bind}}_{\mathsf{F},\mathcal{B}_1,\mathsf{RO}}(\lambda) + \mathsf{Adv}^{\text{po-priv-2}}_{\mathsf{F},\mathcal{B}_2,\mathsf{RO}}(\lambda)\,.$$

*Proof.* We note that the adversary $\mathcal{A}$ has access to the oracles Request and RO. We consider a single intermediate hybrid before reducing the uniqueness to the request privacy against malicious servers (POPRIV2). Let $\mathsf{Adv}_{\mathcal{A},i}(\lambda)$ denote the advantage of $\mathcal{A}$ in $\mathbf{Hybrid_i}$ and let $\mathbf{Hybrid_i} \approx \mathbf{Hybrid_{i+1}}$ denote $|\Pr[\mathbf{Hybrid_i} = 1] - \Pr[\mathbf{Hybrid_{i+1}} = 1]| \leq \mathsf{negl}(\lambda)$.

$\mathbf{Hybrid_0}$: This corresponds to the uniqueness experiment $\mathsf{POUNIQ}_{\mathsf{F},\mathcal{A},\mathsf{RO}}$ (Definition 20). Hence, we have that

$$\mathsf{Adv}^{\text{po-uniq}}_{\mathsf{F},\mathcal{A},\mathsf{RO}}(\lambda) = \mathsf{Adv}_{\mathcal{A},0}(\lambda)\,.$$

$\mathbf{Hybrid_1}$: Let BAD be the event that the adversary queries the Request oracle with two queries $(\mathsf{sk}, \mathsf{aux}, \mathsf{pk}, t, x)$ and $(\mathsf{sk}', \mathsf{aux}', \mathsf{pk}', t', x')$, such that $\mathsf{pk} = \mathsf{pk}'$ and $\mathsf{sk} \neq \mathsf{sk}'$. If BAD happens, then the challenger aborts.

Clearly, we have that $\mathbf{Hybrid_0}$ and $\mathbf{Hybrid_1}$ are identical until the event BAD happens. Hence, we show that we can bound the probability $\Pr[\mathsf{BAD}]$ by constructing a reduction $\mathcal{B}_1$ to the key binding of F. $\mathcal{B}_1$ receives the public parameters pp from the key binding challenger, and forwards pp to $\mathcal{A}$. When $\mathcal{A}$ makes a RO query, then $\mathcal{B}_1$ forwards the same query to its own RO oracle and relays the answer back to $\mathcal{A}$. For answering the Request oracle queries, $\mathcal{B}_1$ proceeds exactly as in $\mathbf{Hybrid_0}$, but when $\mathcal{A}$ submits two queries $(\mathsf{sk}, \mathsf{aux}, \mathsf{pk}, t, x)$ and $(\mathsf{sk}', \mathsf{aux}', \mathsf{pk}', t', x')$ that satisfy $\mathsf{pk} = \mathsf{pk}'$ and $\mathsf{sk} \neq \mathsf{sk}'$, then $\mathcal{B}$ submits the tuple $(\mathsf{pk}, \mathsf{sk}, \mathsf{sk}', \mathsf{aux}, \mathsf{aux}')$ to the key binding challenger. Hence, we have that $\mathcal{B}_1$ wins the key binding experiment with the same probability that the event BAD happens, and therefore, we have that

$$\left|\mathsf{Adv}_{\mathcal{A},0}(\lambda) - \mathsf{Adv}_{\mathcal{A},1}(\lambda)\right| \leq \mathsf{Adv}^{\text{po-key-bind}}_{\mathsf{F},\mathcal{B}_1,\mathsf{RO}}(\lambda)\,,$$

and in particular $\mathbf{Hybrid_0} \approx \mathbf{Hybrid_1}$.

All that remains is to bound the advantage of the adversary $\mathcal{A}$ against the uniqueness experiment in $\mathbf{Hybrid_1}$, i.e., bound $\mathsf{Adv}_{\mathcal{A},1}(\lambda)$. We can do so by constructing a reduction $\mathcal{B}_2$ to the request privacy of $\mathsf{F}$. Analogous to the previous reduction, $\mathcal{B}_2$ receives the public parameters $\mathsf{pp}$ from the request privacy challenger, and forwards $\mathsf{pp}$ to $\mathcal{A}$. Moreover, $\mathsf{RO}$ queries of $\mathcal{A}$ are forwarded to the $\mathsf{RO}$ oracle of $\mathcal{B}_2$ and the response relayed back to $\mathcal{A}$. Upon receiving a $\mathsf{Request}$ query $(\mathsf{sk}, \mathsf{aux}, \mathsf{pk}, t, x)$ from $\mathcal{A}$, $\mathcal{B}_2$ makes a $\mathsf{Request}$ query to its oracle (provided by the request privacy challenger) with the input $(\mathsf{pk}, t, x, x)$, received back a pair of request messages $(\mathsf{req}_0, \mathsf{req}_1)$ and sends $\mathsf{req}_0$ to $\mathcal{A}$. Moreover, $\mathcal{B}_2$ stores the tuple $(\mathsf{pk}, \mathsf{sk}, t, x, \mathsf{req}_1)$ associated with the $\mathsf{Request}$ query.

When $\mathcal{A}$ outputs the responses $(i, j, \mathsf{rep}_{i,0}, \mathsf{rep}_{j,0})$, then $\mathcal{B}_2$ retrieves the tuples $(\mathsf{pk}_i, \mathsf{sk}_i, t_i, x_i, \mathsf{req}_{i,1})$ and $(\mathsf{pk}_j, \mathsf{sk}_j, t_j, x_j, \mathsf{req}_{j,1})$ that it previously stored. We consider the case where $\mathcal{A}$ wins the uniqueness experiment, and hence, in this case we can drop the subscripts as $(\mathsf{pk}_i, \mathsf{sk}_i, t_i, x_i) = (\mathsf{pk}_j, \mathsf{sk}_j, t_j, x_j)$ (note that $\mathsf{sk}_i = \mathsf{sk}_j$ is guaranteed by the key binding property of $\mathsf{F}$ that we previously proved). $\mathcal{B}_2$ calls its $\mathsf{Finalize}$ oracle with $(i, \mathsf{rep}_{i,0}, \mathsf{rep}_{i,1})$, where $\mathsf{rep}_{i,0}$ is provided by $\mathcal{A}$ and $\mathsf{rep}_{i,1}$ is generated honestly by $\mathcal{B}_2$. By the correctness of $\mathsf{F}$, we have that the value returned from the honest flow with $\mathsf{rep}_{i,1}$ is equal to $y := \mathsf{F.Eval}(\mathsf{sk}, t, x)$. If $\mathsf{Finalize}$ oracle returns two different values, then by matching which position $y$ is returned reveals the challenge bit and allows $\mathcal{B}_2$ to win the request privacy experiment. On the other hand, if $\mathsf{Finalize}$ returns two identical values, then $\mathcal{B}_2$ repeats the above for the $j$-th query. If $\mathcal{A}$ wins the uniqueness experiment, then we know that $\mathsf{Finalize}$ will not return identical values for the $j$-th query, and hence, the challenge bit will be revealed, allowing $\mathcal{B}_2$ to win the request privacy experiment. Therefore, $\mathcal{B}_2$ outputs the correct challenge bit in the request privacy experiment, whenever $\mathcal{A}$ wins uniqueness. Hence, we have that

$$\mathsf{Adv}_{\mathcal{A},1}(\lambda) \leq \mathsf{Adv}^{\mathsf{po\text{-}priv\text{-}2}}_{\mathsf{F},\mathcal{B}_2,\mathsf{RO}}(\lambda).$$

Putting everything together, we obtain

$$\mathsf{Adv}^{\mathsf{po\text{-}uniq}}_{\mathsf{F},\mathcal{A},\mathsf{RO}}(\lambda) \leq \mathsf{Adv}^{\mathsf{po\text{-}key\text{-}bind}}_{\mathsf{F},\mathcal{B}_1,\mathsf{RO}}(\lambda) + \mathsf{Adv}^{\mathsf{po\text{-}priv\text{-}2}}_{\mathsf{F},\mathcal{B}_2,\mathsf{RO}}(\lambda),$$

as claimed. This completes the proof of Theorem 8. □

## F Instantiation of the Commitment Scheme

From our analyses in Section 4.1, we can observe that the main properties we need from the commitment scheme are hiding the message (without trapdoor), binding, and enabling extraction of the message using the trapdoor. This is effectively an encryption scheme, and we will employ a variant of Regev's encryption [Reg05] over module lattices for our purposes. Given this Regev-style instantiation is quite well-known by now, we do not delve into too many technical details here and refer the reader to the references provided below for more details. We also note again that for the $\mathbf{d}_x$ commitment to the binary input $x \in \{0,1\}^L$, we can simply employ a standard lattice-based encryption scheme. Therefore, the rest of the section focuses on the other commitments.

For certain instantiations of the message mapping $\mathsf{H}$ (such as BLMR13 PRF [BLMR13]), we need the server's secret key $\mathbf{k}$ to have large coefficients (i.e., unbounded mod $q$). As a result, we need to be able to allow the decryption of large message inputs in the encryption scheme. For this, we can employ the technique (implicit in [GSW13]) described more explicitly in [LNPS21]. Suppose we want to commit to an $N$-dimensional vector over $R_q$, and define $M := N$ if the committed message has small coefficients or $M := 2N$ if the committed message has large coefficients. For readability, we write $\sqrt{q}$ to mean $\lceil \sqrt{q} \rfloor$ below.

- **Setup** samples $\mathbf{A} \xleftarrow{\$} R_q^{n \times (n+M+w)}$. In the case we want to generate the commitment key with a trapdoor, then we set $\mathbf{b}_{i,j}^\top = \mathbf{s}_{i,j}^\top \mathbf{A} + \mathbf{e}_{i,j}^\top$ for $j = 0, 1$ and $i = 1, \ldots, N$, where $\mathbf{s}_{i,j}$'s are $\mathsf{MLWE}$ secret keys where each entry is sampled from a secret distribution $\chi_s$ and $\mathbf{e}_{i,j}$'s are short error vectors where each entry is sampled from an error distribution $\chi_e$. If we don't want the trapdoor, then we simply sample $\mathbf{b}_{i,j}$'s uniformly at random. The commitment key is then set as $\mathsf{ck} = (\mathbf{A}, \mathbf{b}_{1,0}, \mathbf{b}_{1,1}, \ldots, \mathbf{b}_{N,0}, \mathbf{b}_{N,1})$.

– To **Commit** to a message $\mathbf{m} = (m_1, \ldots, m_N) \in R^N$, we proceed as follows:
  1. If $\mathbf{m}$ has *small* coefficients compared to $q$ with $\|m_i\|_\infty \leq q_m$, then we first sample a short randomness vector $\mathbf{r} \leftarrow \chi_e^{n+N+w}$ and set $\mathbf{c} = \mathbf{Ar}$. Then, we compute $c_i = q_m \mathbf{b}_{i,0}^\top \mathbf{r} + m_i$ for $i = 1, \ldots, N$. The function outputs $(\mathbf{c}, c_1, \ldots, c_N) \in R_q^{n+N}$.
  2. If $\mathbf{m}$ has *large* coefficients compared to $q$ (i.e., unbounded mod $q$), then we first sample a short randomness vector $\mathbf{r} \in \chi_e^{n+2N+w}$ and set $\mathbf{c} = \mathbf{Ar}$. Then, we compute $c_{i,0} = \mathbf{b}_{i,0}^\top \mathbf{r} + m_i$ and $c_{i,1} = \mathbf{b}_{i,1}^\top \mathbf{r} + \sqrt{q} m_i$. The function outputs $(\mathbf{c}, c_{1,0}, c_{1,1}, \ldots, c_{N,0}, c_{N,1}) \in R_q^{n+2N}$.

– To **Extract** a message $m_i$ from $c_i$ for $i \in [N]$ using the trapdoor $\mathbf{s}_{i,j}$'s,
  1. If $\mathbf{m}$ has *small* coefficients[14] compared to $q$, we will be given $(\mathbf{c}, c_1, \ldots, c_N) \in R_q^{n+N}$. Then, we compute $u_i := c_i - q_m \mathbf{s}_{i,0}^\top \mathbf{c} = q_m \mathbf{e}_{i,0}^\top \mathbf{r} + m_i \bmod q$. Then, we round off the error term $\mathbf{e}_{i,0}^\top \mathbf{r}$ by computing $u_i \bmod q_m$ to recover each $m_i$.
  2. If $\mathbf{m}$ has *large* coefficients[15] compared to $q$ (i.e., unbounded mod $q$), we will be given $(\mathbf{c}, c_{1,0}, c_{1,1}, \ldots, c_{N,0}, c_{N,1}) \in R_q^{n+2N}$. Then, we compute $u_{i,0} = c_{i,0} - \mathbf{s}_{i,0}^\top \mathbf{c} = \mathbf{e}_{i,0}^\top \mathbf{r} + m_i$, and $u_{i,1} = c_{i,1} - \mathbf{s}_{i,1}^\top \mathbf{c} = \mathbf{e}_{i,1}^\top \mathbf{r} + \sqrt{q} m_i$. Now compute $e' := u_1 - \sqrt{q} u_0 \bmod \sqrt{q}$ and $m_i := (u_1 + e')/\sqrt{q}$.

We note that the server's commitment to the key $\mathbf{k}$ can have small-coefficient message input when BP14 PRF is used; while for BLMR13 PRF, large-coefficient message support is needed. On the other hand, the client's commitment to $(\mathbf{R}, x)$ has always small coefficients in the message (regardless of the PRF) when e.g. the bits of the input message are represented as the coefficients of a polynomial(s).

As already observed in earlier work such as [LNPS21,ESZ22a], we note that the above encryption scheme is effectively an extractable version of the BDLOP commitment scheme [BDL+18]. The (plain) BDLOP commitment is employed for instance in LNP22 proof [LNP22]. Therefore, we can optimize the performance by calculating a single BDLOP commitment (instead of multiple times) when both the underlying NIZK and our OPRF protocol require such a commitment.

As shown in [BDL+18], the commitment scheme is

1. (computationally) hiding[16] if $\mathsf{MLWE}_{n+M,q,w,\chi_e,\chi_s}$ is hard, and
2. (computationally) binding if $\mathsf{MSIS}_{n,n+M+w,2\beta_r}$ is hard where $\beta_r$ denotes the bound on $\|\mathbf{r}\|$ proven by the NIZK proof.

We note that in our case, we prove the commitment opening relations by the client ($\mathbf{c}_r = \mathsf{COM.Commit}(\ldots)$) and the server ($\mathbf{c}_k = \mathsf{COM.Commit}(\ldots)$) *without* a relaxation/approximation factor. Hence, the $\mathsf{MSIS}$ bound $\beta_{\mathrm{SIS}}$ is tighter (compared to what is provided in [BDL+18]) and does not involve terms depending on the size of the relaxation factor. We also note that commitment to $\mathbf{R}$ is done by committing to the rows of $\mathbf{R}$ (under one commitment).

**Extraction/decryption correctness.** In our NIZK proofs employed in $\mathsf{LeOPaRd}$, we prove that all error, randomness, and/or message (if small) coefficients are much smaller than the system modulus $q$. Therefore, the error terms described above to be removed will have a quite small infinity norm compared to the modulus sizes we need for our OPRF. Therefore, extraction/decryption correctness requirements are easily met for well-formed commitments (with probability 1) under typical parameters. Nevertheless, we explicitly state the required bounds below.

1. For *small* messages with $\|m_i\|_\infty \leq q_m$, we need

$$\|q_m \mathbf{e}_{i,0}^\top \mathbf{r} + m_i\|_\infty < q/2.$$

[14] Note that well-formedness of the commitment along with having small coefficients in error, randomness, and message will be proven via a NIZK proof.

[15] Note that well-formedness of the commitment along with "double-encryption" of $(m, \sqrt{q}m)$ will be proven via a NIZK proof. The double-encryption proof often comes for free in communication since it is a simple linear proof over $R_q$ (see, e.g., [LNP22, Section 3]).

[16] In fact, the commitment outputs are (computationally) indistinguishable from uniformly random vectors over $R_q$.

2. For *large* messages, we need

$$\|\mathbf{e}_{i,j}^\top \mathbf{r}\|_\infty < \sqrt{q}/4.$$

To argue that a commitment key with a trapdoor is indistinguishable from a regular commitment key, we require the hardness of $\mathsf{MLWE}_{q,n,\chi_e,\chi_s}$ so that $\mathbf{b}_{i,j}^\top = \mathbf{s}_{i,j}^\top \mathbf{A} + \mathbf{e}_{i,j}^\top$ is indistinguishable from random. For the client's commitment $\mathbf{c}_r$, we assume the error entries are sampled from the same distribution $\chi_r$ as the entries of $\mathbf{R}$ in $\mathsf{LeOPaRd}$ as the client security already relies on $\mathsf{MLWE}$ with its secret dimension $\ell$ and error distribution $\chi_r$. Therefore, we get the dimension parameter $n_c$ for the client's commitment as $n_c = \ell$ (provided the relevant $\mathsf{MSIS}_{n_c,2\beta_r^c}$ problem is hard, which is easily satisfied for our parameter settings). For the server's commitment $\mathbf{c}_k$ as its public key, we assume the error coefficients are sampled from a uniform distribution with standard deviation $\sigma_0$ as the server security already relies on $\mathsf{MLWE}$ with such errors and secret key dimension $m$. Therefore, we get the dimension parameter $n_s$ for the server's commitment as $n_s = m$ (provided the relevant $\mathsf{MSIS}_{n_s,2\beta_r^s}$ problem is hard, which is easily satisfied for our parameter settings).