LeOPaRd: Towards Practical Post-Quantum Oblivious PRFs via Interactive Lattice Problems

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

[1] Faculty of Information Technology, Monash University, Australia
[2] DIENS, École normale supérieure, CNRS, Inria, PSL University, Paris, France
name.surname@monash.edu,erkan.tairi@ens.fr

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 *interactive MLWE with re-use* (iMLWE-RU). We rigorously study the hardness of iMLWE-RU and reduce it (under Gaussian error sampling) to a more standard MLWE-like problem where the adversary is additionally given access to a randomized MLWE PRF oracle. We believe iMLWE-RU problem family can be of independent interest for other interactive protocols.

LeOPaRd exploits an iMLWE-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 136 KB total communication, compared to 300+ KB in earlier works. We also identify gaps in some existing constructions and models, and propose appropriate fixes.

Keywords: Oblivious PRF · Post-Quantum · Lattice · Zero-Knowledge

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 [JKX18].

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 primi-

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

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, $Q_x = 2^{16}$	iMLWE-RU-R + MLWE + MSIS	2	10KB + 126KB	strong, ROM
LeOPaRd, $Q_x = 2^{32}$ (Section 4)		2	20KB + 159KB	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 a one-time offline cost and online cost per query, respectively. We do not consider amortization over batched queries in this table. The server-to-client communication in LeOPaRd, in particular, reduces to just 1-3 KB per query for a batch of 64 (or more) queries (see Table 3). Q denotes a *global* upperbound on the number of OPRF queries; while Q_x denotes the maximum number of OPRF queries *per fixed* PRF tag/input pair (t, x). It is easy to enforce the bound Q_x by counting the number of queries under the same *public* tag t.

tives[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 6, 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 interactive assumption that we introduce, and cleverly combining this with some additional lattice-based techniques that we explain in Section 1.3. We prove our construction secure in the random oracle model in Section 4.2. Specifically, we show that our construction achieves the strong security definitions from [TCR+22]: pseudorandomness, which provides security in the presence of malicious clients (POPRF security), and request privacy against malicious servers (POPRIV2 security). Moreover, in Appendix C.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 1, our construction easily extends to threshold n-out-of-n setting, as in [AG24], thanks to its key-homomorphic property.

Interactive MLWE assumption. We introduce a family of novel interactive assumptions, called *interactive Module Learning with Errors with Re-Use* (iMLWE-RU), and use its rounding-based flavour iMLWE-RU-R 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

[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].

with. This assumption not only allows us to prove the security of our OPRF construction, but also allows us 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 note that our assumption is not "one-more" type (unlike earlier proposals, e.g. [TCR+22, BDFH25, JKK14]), and that we do not make use of the "LimitEval" oracle in the formal model of Tyagi et al. [TCR+22].

The interactive MLWE problem can be seen as a family of assumptions, which roughly state that MLWR-based PRF oracle samples together with MLWE samples are indistinguishable from uniform, even given an oracle that additionally provides "re-used matrix" MLWE samples, where the latter MLWE samples are with respect to a matrix that consists of certain small linear combinations of matrices of existing MLWE samples. Crucially, to circumvent "trapdoor" attacks, the small linear combination coefficients in the "re-used matrix" must be committed by the attacker *before* seeing the original MLWE matrices. This prevents the attacker from meaningfully biasing the "re-used matrix". We therefore intuitively believe that iMLWE-RU is hard, as long as parameters are set such that the "hints" about the original MLWE samples' errors, provided by the "re-used matrix" MLWE oracle, do not reveal much information. To provide formal evidence for this intuition, in Section 3, we provide a reduction from a mild variant of the MLWE problem to iMLWE-RU with discrete Gaussian errors, denoted as iMLWE-RU-G. We highlight that our reduction supports adaptive queries. Our reduction provides evidence regarding the conjectured hardness of the iMLWE-RU assumption, and we encourage more research into investigating its hardness. We note that the iMLWE-RU assumption 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 C.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 C.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. 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] and [HKL+25]. 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 (threshold) 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 OPRF queries, which can potentially hinder the practicality. Our proposal, on the other hand, relies on a bound on the number of queries *per input/tag pair* (x, t), which is easy to check given the tag t is *public*.

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. While this can be done for LNP22 using the Katsumata transform [Kat21] (albeit by incurring a significant performance penalty), it is an open problem how to obtain straight-line extractability for LaBRADOR. We note that 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 $2\times$ 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.

The other 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. It is also based on a heuristic security assumption.

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 $b_x = \mathcal{G}_1(x)$. This term is then blinded via a randomness r and the resulting value $c_x = \mathcal{G}_1(x)^r$ 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 $z = \mathcal{G}_1(x)^k$. The final PRF output is computed via a second hashing as $y = \mathcal{G}_2(x, z)$. 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 $\mathbf{B}_x = \mathcal{G}(x)$ 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 dividing 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 input x and randomness $\mathbf{R}$ *before* seeing the public matrix $\mathbf{A}_r$. To this end, the client first commits to $(\mathbf{R}, x)$, 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 an interactive MLWE assumption. We go beyond intuitive thinking and analyze the security of the construction rigorously. The client's commitment $\mathbf{c}_r$ to $(\mathbf{R}, x)$ can be seen as the client submitting an oracle query with input $(\mathbf{R}, x)$ 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 interactive MLWE-like assumption, iMLWE-RU-R, where the adversary is given access to a SampMLWE-RU oracle that, on input $(\mathbf{R}, x)$ returns a random matrix $\mathbf{A}_r$, an MLWE sample $\mathbf{c}$ under $\mathbf{A}_r$, and an MLWE-like sample $\mathbf{u}$ under $\mathbf{C}_x$ akin to the OPRF paradigm as above. Note that $\mathbf{A}_r$ is re-used in both $\mathbf{c}$ and $\mathbf{u}$, and hence the naming of the assumption. To enable simulation in the pseudorandomness analysis, the adversary is also given access to a Samp-PRF oracle that outputs a rounding sample $\lfloor \mathbf{B}_x\mathbf{k} \rceil_p$ for a given x. Due to this rounding error used in Samp-PRF, we call this iMLWE-RU flavour as iMLWE-RU-R.

Security reduction for iMLWE-RU-G from MLWE-PRF. We define another flavour of iMLWE-RU, denoted as iMLWE-RU-G, where the above rounding error in the underlying lattice-based PRF oracle Samp-PRF is replaced by a random discrete Gaussian error of the same standard deviation (see Section 3 for details). We believe this minor change does not impact practical hardness of the problem, which should stem mainly from the size of the error rather than the exact shape of the PRF error distribution (we remark that it is common for reductions between lattice problems to rely on Gaussian distributions while the higher-level schemes are instantiated with another distribution. For example, NIST-standardized Dilithium [DKL+18] does not use any Gaussian distributions).

[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.

We then give a hardness reduction for iMLWE-RU-G from a mild variant of MLWE that we call MLWE-PRF. The MLWE-PRF assumption is that the randomized function $x \mapsto \mathcal{G}(x)\cdot\mathbf{k}+\mathbf{e}$, where $\mathbf{e}$ is a small error following a Gaussian distribution is pseudorandom even when given additional MLWE samples for $\mathbf{k}$ with respect to uniformly random matrices $\mathbf{A}_r$. Therefore, the MLWE-PRF assumption is a mild (randomized) variant of the pseudorandomness of the MLWR-based PRF $x \mapsto \lfloor \mathcal{G}(x) \cdot \mathbf{k} \rceil_p$, which has been proved based on the MLWE assumption when $\mathcal{G}$ is implemented appropriately as in [BLMR13] and [BP14].

In the iMLWE-RU-G problem, each of the attacker's oracle queries $(x, \{\mathbf{R}_j\}_j)$ is answered with MLWE samples $(\mathbf{B}_x, \mathbf{y}_B = \mathbf{B}_x\mathbf{k} + \mathbf{e}_B)$ and $(\mathbf{A}_{r,j}, \mathbf{c}_j = \mathbf{A}_{r,j}\mathbf{k} + \mathbf{e}_j)$ where $\mathbf{B}_x := \mathcal{G}(x)$, with respect to "fresh" matrices along with a "re-used" matrix MLWE samples of the form $(\mathbf{C}_{x,j} := \mathbf{R}_j\mathbf{A}_{r,j} + \mathbf{B}_x, \mathbf{u}_j = \mathbf{C}_{x,j}\mathbf{k} + \bar{\mathbf{e}}'_j)$, reminiscent of the "Reused-A LWE" [MS25] and Hint-MLWE problem [KLSS23]. Accordingly, our security reduction from MLWE-PRF to iMLWE-RU-G proceeds in two steps. In the first step, we reduce a variant of the Hint-MLWE problem called HintMLWE-PRF to iMLWE-RU-G. In the HintMLWE-PRF problem, besides the MLWE samples w.r.t. $\mathbf{B}_x$ and $\mathbf{A}_{r,j}$ matrices, the adversary is also provided with appropriate linear combinations of the error vectors as hints of the form $\mathbf{h}_j := \mathbf{R}_j\mathbf{e}_j - \mathbf{e}'_j$ and $\mathbf{h}_{B,j} := \mathbf{e}_B - \mathbf{e}'_{B,j}$. These hints allow the reduction from MLWE-PRF to iMLWE-RU-G to simulate the 're-used' samples as $\mathbf{u}_j = \mathbf{R}_j\mathbf{c}_j + \mathbf{y}_B - (\mathbf{h}_j + \mathbf{h}_{B,j}) = \mathbf{C}_{x,j}\mathbf{k} + \mathbf{e}'_j + \mathbf{e}'_{B,j}$, where $\bar{\mathbf{e}}'_j := \mathbf{e}'_j + \mathbf{e}'_{B,j}$ is the simulated error term for $\mathbf{u}_j$. We remark that this reduction could also directly use a single combined hint of the form $\bar{\mathbf{h}}_j := \mathbf{h}_j + \mathbf{h}_{B,j} = \mathbf{R}_j\mathbf{c}_j + \mathbf{y}_B - \bar{\mathbf{e}}'_j$; however such a combined hint variant of Hint-MLWE makes it difficult to deal with adaptive oracle $\mathbf{R}_j$ queries in the following reduction from MLWE to our HintMLWE-PRF problem in the second step below, hence we chose to split the hint into the two separate hints $\mathbf{h}_j$ and $\mathbf{h}_{B,j}$). Note that in HintMLWE-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 HintMLWE-PRF can be viewed as a variant of the Matrix Hint-MLWE [ENP24] that generalizes Hint-MLWE [KLSS23] to the case of hint matrices $\mathbf{R}_j$ for the hints $\mathbf{h}_j$, while our HintMLWE-PRF variant also includes a PRF 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 HintMLWE-PRF by adapting the MLWE to Matrix Hint-MLWE [ENP24] reduction.

Overall, we achieve a reduction from MLWE-PRF to iMLWE-RU-G, provided that the necessary parameter constraints are satisfied. Our 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 iMLWE-RU-G. In particular, our reduction requires the "masking ratio" lower bound $\sigma_1/\sigma \geq \sqrt{B_r} \approx \sqrt{Q_x^\infty}$ for large Q_x^∞, where σ_1 and σ denote the width (std dev) of the masking errors $\mathbf{e}'_j$ and MLWE errors $\mathbf{e}_j$, respectively, and Q_x^∞ 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 the following simple error "averaging" attack on iMLWE-RU-G (and our OPRF protocol as well as those using the same blueprint [ADDS21, AG24]). In this attack, the adversary uses the Q_x^∞ 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 $c > 1$, and more generally, a necessary condition to prevent the averageing attack from reducing the error standard deviation (and MLWE security) below the σ 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 iMLWE-RU-G security reduction parameter lower bound discussed above (for large Q_x^∞), 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.

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 A on input y, outputs x. When we want to make the randomness r used by A explicit, we write it as $x \leftarrow \mathsf{A}(y; r)$. If A has oracle access to a procedure O, we represent this by superscript A^{O}. If 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 σ composed of two elements σ_1 and σ_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 denote by $R_q = \mathbb{Z}_q[X]/(X^d + 1)$. We write R_{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 the result 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 ∞-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 “$\Game(x)$; ꓘ(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.

2.1 (Partially) Oblivious PRF

We use the game-based definitions given in [TCR+22] and [ADDG24]. An OPRF is a protocol between a server S that has a private key k and a client 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 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 [TCR+22]). *A partial oblivious PRF (POPRF)* F *is a tuple of* PPT *algorithms*

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

The Setup *algorithm and the* KeyGen *algorithm generate public parameters* pp *and a public/secret key pair* $(\mathsf{pk}, \mathsf{sk})$*, respectively. Oblivious evaluation is carried out as an interactive protocol between* C *and* S*, presented as algorithms* $\mathsf{F.Request}, \mathsf{F.BlindEval}, \mathsf{F.Finalize}$*, which work as follows (where the random oracles are denoted by* RO*):*

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

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

```
$\mathsf{POPRF}^b_{\mathsf{F},\mathcal{A},\mathcal{S},\mathsf{RO}}(\lambda)$
$q_{t,s} := 0, q_t := 0$
$\mathsf{st}_\mathsf{F} \leftarrow \mathsf{RO}_\mathsf{F}.\mathsf{Init}(1^\lambda)$   $/\!\!/\ \mathcal{T} \times \mathcal{X} \rightarrow \mathcal{Z}$
$\mathsf{st}_\mathsf{RO} \leftarrow \mathsf{RO}.\mathsf{Init}(1^\lambda)$
$\mathsf{pp}_1 \leftarrow \mathsf{F}.\mathsf{Setup}(1^\lambda)$
$(\mathsf{st}_\mathcal{S}, \mathsf{pk}_0, \mathsf{pp}_0) \leftarrow \mathcal{S}.\mathsf{Init}(\mathsf{pp}_1)$
$(\mathsf{sk}, \mathsf{pk}_1) \leftarrow \mathsf{F}.\mathsf{KeyGen}^{\mathsf{RO}}_{\mathsf{pp}_1}(1^\lambda)$
$b' \leftarrow \mathcal{A}^{\mathsf{Eval},\mathsf{BlindEval},\mathsf{Prim}}(\mathsf{pp}_b, \mathsf{pk}_b)$
return $b'$

$\mathsf{BlindEval}(t, \mathsf{req})$
$q_t := q_t + 1$
$(\mathsf{rep}_0, \mathsf{st}_\mathcal{S}) \leftarrow \mathcal{S}.\mathsf{BlindEval}^{\mathsf{LimitEval}}(t, \mathsf{req}, \mathsf{st}_\mathcal{S})$
$\mathsf{rep}_1 \leftarrow \mathsf{F}.\mathsf{BlindEval}^{\mathsf{RO}}_{\mathsf{pp}_1}(\mathsf{sk}, t, \mathsf{req})$
return $\mathsf{rep}_b$
```

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

$\mathsf{LimitEval}(t, x)$
$q_{t,s} := q_{t,s} + 1$
if $q_{t,s} \leq q_t$ then
    return $\mathsf{Eval}(t, x)$
return $\perp$

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

```
$\mathsf{POPRIV2}^b_{F,\mathcal{A},\mathsf{RO}}(\lambda)$
$\mathsf{pp} \leftarrow \mathsf{F}.\mathsf{Setup}(1^\lambda)$
$i := 0$
$b' \leftarrow \mathcal{A}^{\mathsf{Request},\mathsf{Finalize},\mathsf{RO}}(\mathsf{pp})$
return $b'$

$\mathsf{Request}(\mathsf{pk}, t, x_0, x_1)$
$i := i + 1$
$(\mathsf{st}_{i,0}, \mathsf{req}_0) \leftarrow \mathsf{F}.\mathsf{Request}^{\mathsf{RO}}_{\mathsf{pp}}(\mathsf{pk}, t, x_0)$
$(\mathsf{st}_{i,1}, \mathsf{req}_1) \leftarrow \mathsf{F}.\mathsf{Request}^{\mathsf{RO}}_{\mathsf{pp}}(\mathsf{pk}, t, x_1)$
return $(\mathsf{req}_b, \mathsf{req}_{1-b})$
```

```
$\mathsf{Finalize}(j, \mathsf{rep}_0, \mathsf{rep}_1)$
if $j > i$ then
    return $\perp$
$y_b \leftarrow \mathsf{F}.\mathsf{Finalize}^{\mathsf{RO}}_{\mathsf{pp}}(\mathsf{rep}_0, \mathsf{st}_{j,b})$
$y_{1-b} \leftarrow \mathsf{F}.\mathsf{Finalize}^{\mathsf{RO}}_{\mathsf{pp}}(\mathsf{rep}_1, \mathsf{st}_{j,1-b})$
if $y_0 = \perp \vee y_1 = \perp$ then
    return $\perp$
return $(y_0, y_1)$
```

Fig. 1: POPRF and POPRIV2 Experiments.

We also define 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 use the correctness definition from [ADDG24], which allows for a small failure probability, and then define *pseudorandomness* against malicious clients as in [TCR+22].

Definition 2 (Correctness [ADDG24]). *A partial oblivious PRF (POPRF)* F *is correct, if the following holds for* $\mathsf{pp} \leftarrow \mathsf{F}.\mathsf{Setup}(1^\lambda)$ *and* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{F}.\mathsf{KeyGen}^{\mathsf{RO}}_{\mathsf{pp}}(1^\lambda)$

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

Definition 3 (Pseudorandomness [TCR+22]). *We say that a partial oblivious PRF (POPRF)* F *is pseudorandom if for all* PPT *adversaries* $\mathcal{A}$*, there exists a* PPT *simulator* $\mathcal{S}$ *and a negligible function* $\mathsf{negl}(\cdot)$*, such that the following holds,*

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

where the experiment POPRF *is defined in Figure 1.*

The oracle Prim captures access to the random oracle(s) used in the POPRF construction. For $b = 0$ (i.e., when the adversary interacts with a simulator and a truly random function), the simulator may only use a limited number of random function queries to simulate the random oracle accessed via Prim.

Lastly, we define the *request privacy*, which provides security against malicious servers. We present the stronger POPRIV2 definition satisfied by our proposal, and refer the reader to [TCR+22] for the weaker POPRIV1 definition.

Definition 4 (Request Privacy [TCR+22]). *We say that a POPRF* F *has request privacy against malicious servers if for all* PPT *adversaries, there exists a negligible function* $\mathsf{negl}(\cdot)$ *upper-bounding the following,*

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

where the experiment $\mathsf{POPRIV2}$ *is defined in Figure 1.*

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* ϵ *in solving* $\mathsf{MSIS}_{n,m,B}$*.*

Definition 6 ($\mathsf{MLWE}_{n,m,\chi}$)**.** *Let* χ *be an error distribution over* R*,* $\mathbf{A} \xleftarrow{\$} R_q^{m\times n}$*, where* m*,* $n > 0$*, let* $\mathbf{s} \xleftarrow{\$} \chi^n$ *be a secret vector and* $\mathbf{e} \xleftarrow{\$} \chi^m$ *be an error vector. The* $\mathsf{MLWE}_{n,m,\chi}$ *problem asks an adversary* $\mathcal{A}$ *to distinguish between* $(\mathbf{A}, \mathbf{As}+\mathbf{e})$ *and* $(\mathbf{A}, \mathbf{b})$*, for* $\mathbf{b} \xleftarrow{\$} R_q^m$*. We say* $\mathcal{A}$ *has advantage* ϵ *against the* $\mathsf{MLWE}_{n,m,\chi}$ *problem if*

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

Definition 7 ($\mathsf{knMLWE}_{n,m,h,\chi}$)**.** *Let* χ *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* ϵ *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 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 MSIS, MLWE, and 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*[7] *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 Λ 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* PPT *algorithm that, given a basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a full rank n-dimensional lattice Λ, 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 Λ 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 Λ^* denotes the dual lattice of Λ. 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 Λ, 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 Λ.

Lemma 4 ([KLSS23]). *For $n \in \mathbb{Z}^+, \epsilon \in \mathbb{R}^+$, n dimensional lattice Λ, 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 [KLSS23, Lemma 7]). *Fix $s_1 \in \mathbb{R}^+$, $n, \ell \in \mathbb{Z}^+$, a positive definite symmetric matrix $\mathbf{\Sigma}_1$, and a matrix $\bar{\mathbf{R}} \in \mathbb{Z}^{\ell\times n}$. Furthermore, let $\mathbf{\Sigma}_0 := \left(\mathbf{\Sigma}_1^{-1} + \frac{1}{s_1^2}\bar{\mathbf{R}}^\top\bar{\mathbf{R}}\right)^{-1}$. Then, the following two probability distributions over $\mathbb{Z}^{n+\ell}$ are equal:*

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

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

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

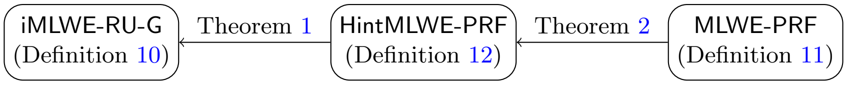

Fig. 2: Summary of the reductions among our iMLWE-RU-G and other lattice assumptions. Solid arrows represent the reductions given in this work.

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

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

where $\mathbf{c} := \frac{1}{s_1^2}\mathbf{\Sigma}_0\bar{\mathbf{R}}^\top\mathbf{w}$. 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 $\Pr_{(\tilde{\mathbf{e}},\mathbf{h}) \xleftarrow{\$} \mathcal{D}_2}[\tilde{\mathbf{e}} = \mathbf{v} | \mathbf{h} = \mathbf{w}]$ 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 iMLWE-RU: New Interactive Lattice Problem

We introduce a family of new interactive MLWE assumptions, and particularly focus on two flavours: (i) iMLWE-RU-G uses discrete Gaussian noise in its PRF oracle, and (ii) iMLWE-RU-R uses rounding in its PRF oracle. Since the problems are identical otherwise, we present iMLWE-RU-G here in the main body and refer to Appendix B.1 for the formal definition of iMLWE-RU-R.

We prove a set of reductions to establish evidence for the conjectured hardness of iMLWE-RU-G. To this end, we first introduce an intermediary problem, HintMLWE-PRF, and show a reduction from HintMLWE-PRF to iMLWE-RU-G. Here, we interpret the error information leaked in iMLWE-RU-G as additional hints (hence the name of the problem). Then, we introduce the main problem that form the basis for the conjectured hardness of iMLWE-RU-G, namely MLWE-PRF. This problem is a mild variant of the MLWE problem, where the adversary is additionally given PRF outputs. We show a reduction from MLWE-PRF to HintMLWE-PRF, which overall leads a reduction from MLWE-PRF to iMLWE-RU-G. Summary of the reductions that we achieve is pictorially depicted in Figure 2.

Definition 10 **(iMLWE-RU-G$_{\mathcal{G},Q,Q_x^\infty,q,m,N,h,L,\beta_r,\bar{\chi}}$).** *Let $\bar{\chi} = (\chi, \chi_B, \chi_1, \chi_k)$ be discrete distributions over R, let $\mathcal{G}$ be a hash family, and let $Q, Q_x^\infty, q, m, N, h, L, \beta_r \in \mathbb{Z}^+$. We say that the interactive MLWE assumption* iMLWE-RU-G$_{\mathsf{param}}$ *holds, for* $\mathsf{param} := (\mathcal{G}, Q, Q_x^\infty, q, m, N, h, L, \beta_r, \bar{\chi})$*, if for all* PPT *adversaries $\mathcal{A}$, there exists a negligible function* $\mathsf{negl}(\cdot)$*, such that the following holds,*

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

where the experiment iMLWE-RU-G^b *is defined in Figure 3, Q_x denotes the number of MLWE sample matrix pairs $(\mathbf{A}_{r,j}, \mathbf{C}_{x,j})_{j \in [Q_x]}$ requested from the oracle for a queried x, $Q := \sum_x Q_x$ denotes the total number of requested matrix pairs in the experiment, and Q_x^∞ is the maximum allowed upper bound on $\max_x Q_x$.*

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

```
iMLWE-RU-G$^b_{\mathcal{G},Q,Q_x^\infty,q,m,N,h,L,\beta_r,\bar{\chi}}(\lambda)$
$\mathbf{k} \xleftarrow{\$} \chi_k^m$
$b' \leftarrow \mathcal{A}^{\mathsf{SampMLWE\text{-}RU}^b(\cdot,\cdot),\mathsf{SampMLWE\text{-}PRF}^b(\cdot)}(1^\lambda)$
return $b = b'$
```

```
SampMLWE-RU$^b(\mathbf{R} \in R^{h \times N}, x \in \{0,1\}^L)$
if $\|\mathbf{R}\|_\infty > \beta_r$ then return $\perp$
$\mathbf{A}_r \xleftarrow{\$} R_q^{N \times m}, \mathbf{e} \xleftarrow{\$} \chi^N$
$\mathbf{B}_x = \mathcal{G}(x) \in R_q^{h \times m}$
$\mathbf{C}_x = \mathbf{R}\mathbf{A}_r + \mathbf{B}_x \in R_q^{h \times m}$, $\mathbf{e}' \xleftarrow{\$} \chi_1^h$
if $\mathcal{Z}[x] = \perp$ then SampMLWE-PRF$^0(x)$
$\mathbf{z}' := \mathcal{Z}[x]$
$\widetilde{\mathbf{e}}' \xleftarrow{\$} \chi_1^h$, $\widetilde{\mathbf{e}} \xleftarrow{\$} \chi^N$, $\widetilde{\mathbf{e}}_\mathbf{B} \xleftarrow{\$} \chi_B^h$   // simulated errors
$\mathbf{e}_{\mathsf{sim}} = \widetilde{\mathbf{e}}' - \widetilde{\mathbf{e}}_\mathbf{B} - \mathbf{R}\widetilde{\mathbf{e}} \in R^h$
if $b = 0$ then
    $\mathbf{c} \xleftarrow{\$} R_q^N$, $\mathbf{u} = \mathbf{R}\mathbf{c} + \mathbf{z}' + \mathbf{e}_{\mathsf{sim}} \in R_q^h$
if $b = 1$ then
    $\mathbf{c} = \mathbf{A}_r\mathbf{k} + \mathbf{e} \in R_q^N$, $\mathbf{u} = \mathbf{C}_x\mathbf{k} + \mathbf{e}' \in R_q^h$
return $(\mathbf{A}_r, \mathbf{c}, \mathbf{u})$
```

```
SampMLWE-PRF$^b(x)$
if $b = 0$ then
    if $\mathcal{Z}[x] = \perp$ then
        $\mathbf{z}' \xleftarrow{\$} R_q^h, \mathcal{Z}[x] := \mathbf{z}'$
if $b = 1$ then
    if $\mathcal{Z}[x] = \perp$ then
        $\mathbf{B}_x = \mathcal{G}(x) \in R_q^{h \times m}$
        $\mathbf{e}_B \xleftarrow{\$} \chi_B^h$
        $\mathbf{y}_B = \mathbf{B}_x\mathbf{k} + \mathbf{e}_B \in R_q^h$
        $\mathcal{Z}[x] := \mathbf{y}_B$
$\mathbf{y}_B = \mathcal{Z}[x]$
return $\mathbf{y}_B$
```

Fig. 3: iMLWE-RU-G security game.

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

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

where the experiment MLWE-PRF^b *is defined in Figure 4, Q_P denotes the number of queries to the MLWE PRF oracle* SampMLWE-PRF^b*, while Q_M denotes the number of queries to the MLWE oracle* SampMLWE^b*.*

We present our reduction from MLWE-PRF to iMLWE-RU-G via an intermediate problem, which is a "hint" variant of the iMLWE-RU-G problem, where instead of the re-used MLWE samples we provide the adversary with MLWE hints of the form $\mathbf{h}_i := \mathbf{R}_i\mathbf{e}_i + \mathbf{e}'_i$ for the MLWE error $\mathbf{e}'_i$ and hints of the form $\mathbf{h}_{B,i} := \mathbf{e}_B + \mathbf{e}'_{B,i}$ for the PRF-MLWE error $\mathbf{e}_B$. We call the latter variant of iMLWE-RU-G the HintMLWE-PRF problem. Our HintMLWE-PRF problem can be viewed as a special case 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], to allow for general hint matrices, rather than block diagonal hint matrices used in [KLSS23].

Definition 12 (HintMLWE-PRF$_{\mathcal{G},Q,Q_x^\infty,q,m,N,h,L,\beta_r,\bar{\chi}}$)**.** *Let $\bar{\chi} = (\chi, \chi_B, \chi_1, \chi_{1,B}, \chi_k)$ be discrete distributions over R, let $\mathcal{G}$ be a hash family, and let $Q, Q_x^\infty, q, m, N, h, L, \beta_r \in \mathbb{Z}^+$. We say that the* HintMLWE-PRF$_\mathsf{param}$ *holds, for* param $:= (\mathcal{G}, Q, Q_x^\infty, q, m, N, h, L, \beta_r, \bar{\chi})$*, if for all* PPT *adversaries $\mathcal{A}$,there exists a negligible function* $\mathsf{negl}(\cdot)$*, such that the following holds,*

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

```
$\mathsf{MLWE\text{-}PRF}_{\mathcal{G},Q_\mathsf{P},Q_\mathsf{M},q,m,h,L,\bar{\chi}}(\lambda)$
$\mathbf{k} \xleftarrow{\$} \chi_k^m$
$b' \leftarrow \mathcal{A}^{\mathsf{SampMLWE\text{-}PRF}^b(\cdot),\mathsf{SampMLWE}^b(\cdot)}(1^\lambda)$
**return** $b = b'$
```

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

```
$\mathsf{SampMLWE\text{-}PRF}^b(x)$
**if** $b = 0$ **then**
    **if** $\mathcal{Z}[x] = \perp$ **then**
        $\mathbf{z}' \xleftarrow{\$} R_q^h, \mathcal{Z}[x] \coloneqq \mathbf{z}'$
**if** $b = 1$ **then**
    **if** $\mathcal{Z}[x] = \perp$ **then**
        $\mathbf{B}_x = \mathcal{G}(x) \in R_q^{h \times m}$
        $\mathbf{e}_B \xleftarrow{\$} \chi_B^h$
        $\mathbf{y}_B = \mathbf{B}_x \mathbf{k} + \mathbf{e}_B \in R_q^h$
        $\mathcal{Z}[x] \coloneqq \mathbf{y}_B$
$\mathbf{y}_B = \mathcal{Z}[x]$
**return** $\mathbf{y}_B$
```

Fig. 4: MLWE-PRF security game

where the experiment HintMLWE-PRF^b *is defined in Figure 5,* Q_x *denotes the number of MLWE sample matrix/hint pairs* $(\mathbf{A}_{r,j}, \mathbf{h}_j)_{j \in [Q_x]}$ *requested from the oracle for a queried* x, $Q \coloneqq \sum_x Q_x$ *denotes the total number of requested hint vectors during the experiment, and* Q_x^∞ *is the maximum allowed upper bound on* $\max_x Q_x$.

3.1 Hardness Reduction from HintMLWE-PRF to iMLWE-RU-G

As an intermediate conceptual step in our reduction from MLWE-PRF to iMLWE-RU-G, we observe that we can reduce a "hint" variant HintMLWE-PRF of MLWE-PRF (inspired by the Hint-MLWE problem [KLSS23]) to iMLWE-RU-G. The idea is that in the HintMLWE-PRF problem, in addition to the MLWE-PRF samples $(\mathbf{B}_x, \mathbf{y}_B = \mathbf{B}_x\mathbf{k} + \mathbf{e}_B)$ and $(\mathbf{A}_r, \mathbf{c} = \mathbf{A}_r\mathbf{k} + \mathbf{e})$, the adversary is also provided with appropriate matrix hint $\mathbf{h}_j$ about error $\mathbf{e}$ with respect to the $\mathbf{R}$ matrix provided by the adversary, i.e. $\mathbf{h} \coloneqq \mathbf{Re} - \mathbf{e}'$, and also a hint $\mathbf{h}_B \coloneqq \mathbf{e}_B - \mathbf{e}'_B$. The hints $\mathbf{h}$ and $\mathbf{h}_B$ then allow the adversary to simulate the "re-used" iMLWE-RU-G samples of the form $(\mathbf{C}_x \coloneqq \mathbf{RA}_r + \mathbf{B}_x, \mathbf{u} = \mathbf{C}_x\mathbf{k} + \bar{\mathbf{e}}')$ from the given MLWE-PRF samples, since $\mathbf{Rc} + \mathbf{y}_B - (\mathbf{h} + \mathbf{h}_B) = \mathbf{C}_x\mathbf{k} + \mathbf{e}' + \mathbf{e}'_B$, where $\bar{\mathbf{e}}'_j \coloneqq \mathbf{e}' + \mathbf{e}'_B$ is the simulated error term for $\mathbf{u}_j$, and by the discrete Gaussian convolution Lemma, is statistically close to a discrete Gaussian with parameter $\bar{s}_1 \coloneqq \sqrt{s_1^2 + s_{1,B}^2}$ when the independent errors $\mathbf{e}'$ and $\mathbf{e}'_B$ are discrete Gaussians with parameters s_1 and $s_{1,B}$ respectively.

Theorem 1. *Let* $\bar{\chi} \coloneqq (\chi, \bar{\chi}_B, \bar{\chi}_1, \chi_k)$ *be discrete distributions over* R, *let* $\mathcal{G}$ *be a hash family, and let* $Q, Q_x^\infty, q, m, N, h, L, \beta_r \in \mathbb{Z}^+, \epsilon \in \mathbb{R}^+$. *Suppose that* $\chi_B \coloneqq \mathcal{D}_{\mathbb{Z}^d, s_B}$, $\bar{\chi}_B \coloneqq \mathcal{D}_{\mathbb{Z}^d, \bar{s}_B}$ *with* $\bar{s}_B \geq s_B$. *Moreover, let* $\bar{\chi}_h = (\chi, \chi_B, -\chi_1, -\chi_{1,B}, \chi_k)$, *where for any distribution* χ, *we denote by* $-\chi$ *the distribution obtained by sampling* e' *from* χ *and outputting* $-e'$. *The* iMLWE-RU-G$_{\mathcal{G},Q,Q_x^\infty,q,m,N,h,L,\beta_r,\bar{\chi}}$ *assumption holds if the* HintMLWE-PRF$_{\mathcal{G},Q,Q_x^\infty,q,m,N,h,L,\beta_r,\bar{\chi}_h}$ *assumption holds. More precisely, for any* PPT *adversary* $\mathcal{A}$ *against the* iMLWE-RU-G *problem, there exists a* PPT *adversary* $\mathcal{B}$ *against the* HintMLWE-PRF *problem, such that*

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

where $\bar{\epsilon} = 0$ *if* $\bar{\chi}_1 \coloneqq \chi_1 + \chi_{1,B}$ *is the convolution of the distributions* χ_1 *and* $\chi_{1,B}$. *Moreover, if* $\chi_1 \coloneqq \mathcal{D}_{\mathbb{Z}^d, s_1}$, $\chi_{1,B} \coloneqq \mathcal{D}_{\mathbb{Z}^d, s_{1,B}}$ *and* $\bar{\chi}_1 \coloneqq \mathcal{D}_{\mathbb{Z}^d, \bar{s}_1}$ *with* $\bar{s}_1 \coloneqq \sqrt{s_1^2 + s_{1,B}^2}$, *and* $\min(\bar{s}_1, \bar{s}_B) \geq \sqrt{\ln(2hd(1 + 1/\epsilon))/\pi}$, *then the above advantage bound holds with* $\bar{\epsilon} = 4Q \cdot \epsilon$.

```
$\mathsf{HintMLWE\text{-}PRF}^b_{\mathcal{G},Q,Q_x^\infty,q,m,N,h,L,\beta_r,\bar{\chi}}(\lambda)$
$\mathbf{k} \xleftarrow{\$} \chi_k^m$
$b' \leftarrow \mathcal{A}^{\mathsf{SampHMLWEP}^b(\cdot,\cdot),\mathsf{SampMLWE\text{-}PRF}^b(\cdot)}(1^\lambda)$
$\mathbf{return}\ b = b'$

$\mathsf{SampHMLWEP}^b(\mathbf{R} \in R^{h\times N}, x \in \{0,1\}^L)$
$j[x] := j[x]+1, j := j[x], \mathbf{R}_j := \mathbf{R}$
$\mathbf{if}\ \|\mathbf{R}_j\|_\infty > \beta_r$
 $\mathbf{then\ return}\ \perp$
$\mathbf{B}_x = \mathcal{G}(x) \in R_q^{h\times m}$
$\mathbf{y}_B := \mathsf{SampMLWE\text{-}PRF}^b(x)$
$\mathbf{e}_B = \mathcal{E}[x]$
$\mathbf{A}_{r,j} \xleftarrow{\$} R_q^{N\times m}$
$\mathbf{e}'_j \xleftarrow{\$} \chi_1^h, \mathbf{e}_j \xleftarrow{\$} \chi^N, \mathbf{e}'_{B,j} \xleftarrow{\$} \chi_{1,B}^h$    // real errors for $b = 1$
$\widetilde{\mathbf{e}'_j} \xleftarrow{\$} \chi_1^h, \widetilde{\mathbf{e}_j} \xleftarrow{\$} \chi^N, \widetilde{\mathbf{e}'_{B,j}} \xleftarrow{\$} \chi_{1,B}^h$    // simulated errors for $b = 0$
$\mathbf{if}\ b = 0\ \mathbf{then}$
    $\mathbf{c}_j \xleftarrow{\$} R_q^N, \mathbf{h}_j = \mathbf{R}_j\widetilde{\mathbf{e}_j} + \widetilde{\mathbf{e}'_j}, \mathbf{h}_{B,j} = \widetilde{\mathbf{e}_B} + \widetilde{\mathbf{e}'_{B,j}}$
$\mathbf{if}\ b = 1\ \mathbf{then}$
    $\mathbf{c}_j = \mathbf{A}_{r,j}\mathbf{k} + \mathbf{e}_j \in R_q^N, \mathbf{h}_j = \mathbf{R}_j\mathbf{e}_j + \mathbf{e}'_j, \mathbf{h}_{B,j} = \mathbf{e}_B + \mathbf{e}'_{B,j}$
$\mathbf{return}\ ((\mathbf{A}_{r,j}, \mathbf{c}_j, \mathbf{h}_j), (\mathbf{B}_x, \mathbf{y}_B, \mathbf{h}_{B,j}))$
```

```
$\mathsf{SampMLWE\text{-}PRF}^b(x)$
$\mathbf{if}\ b = 0\ \mathbf{then}$
    $\mathbf{if}\ \mathcal{Z}[x] = \perp\ \mathbf{then}$
        $\widetilde{\mathbf{e}_B} \xleftarrow{\$} \chi_B^h, \mathcal{E}[x] := \widetilde{\mathbf{e}_B}$
        $\mathbf{z}' \xleftarrow{\$} R_q^h, \mathcal{Z}[x] := \mathbf{z}'$
$\mathbf{if}\ b = 1\ \mathbf{then}$
    $\mathbf{if}\ \mathcal{Z}[x] = \perp\ \mathbf{then}$
        $\mathbf{B}_x = \mathcal{G}(x) \in R_q^{h\times m}$
        $\mathbf{e}_B \xleftarrow{\$} \chi_B^h, \mathcal{E}[x] := \mathbf{e}_B$
        $\mathbf{y}_B = \mathbf{B}_x\mathbf{k} + \mathbf{e}_B \in R_q^h$
        $\mathcal{Z}[x] := \mathbf{y}_B$
$\mathbf{y}_B = \mathcal{Z}[x]$
$\mathbf{return}\ \mathbf{y}_B$
```

Fig. 5: HintMLWE-PRF security game.

Proof. Given an adversary $\mathcal{A}$ against iMLWE-RU-G, we construct an adversary $\mathcal{B}$ against HintMLWE-PRF that runs as shown in Figure 6.

Next, to analyze the advantage of $\mathcal{B}$ against HintMLWE-PRF, we consider first the case that $b = 1$. In this case, for a query $(\mathbf{R}, x)$ of $\mathcal{A}$ to $\mathsf{SimSampMLWE\text{-}RU}^b$ the response $((\mathbf{A}_r, \mathbf{c}, \mathbf{h}), (\mathbf{B}_x, \mathbf{y}_B, \mathbf{h}_B))$ returned by the $\mathsf{SampHMLWEP}^b$ oracle to query $(\mathbf{R}, x)$ has $\mathbf{y}_B = \mathbf{B}_x\mathbf{k} + \mathbf{e}_B$ with $\mathbf{B}_x = \mathcal{G}(x)$, $\mathbf{c} = \mathbf{A}_r\mathbf{k} + \mathbf{e} \in R_q^N$ with hints $\mathbf{h} = \mathbf{R}\mathbf{e} + \mathbf{e}'$ and $\mathbf{h}_B = \mathbf{e}_B + \mathbf{e}''_B$, and $\mathbf{e}'$ (resp. $\mathbf{e}''_B$) sampled from $-\chi_1$ (resp. $-\chi_{1,B}$). It follows that $\mathbf{u} = \mathbf{R}\mathbf{c} + \mathbf{y}'_B - (\mathbf{h} + \mathbf{h}_B) = \mathbf{R}\mathbf{c} + \mathbf{y}_B - (\mathbf{h} + \mathbf{h}_B + \mathbf{e}'_B) = \mathbf{C}_x\mathbf{k} + \bar{\mathbf{e}}'$ with $\mathbf{C}_x := \mathbf{R}_j\mathbf{A}_{r,j} + \mathbf{B}_x$ and $\bar{\mathbf{e}}' := -(\mathbf{e}' + \mathbf{e}''_B)$ is sampled from the distribution $\chi_1 + \chi_{1,B}$. Therefore, in the $\mathsf{HintMLWE\text{-}PRF}^1$ game, the view of $\mathcal{A}$ is simulated with exactly the same distribution as in the real $\mathsf{iMLWE\text{-}RU\text{-}G}^1$ game with $\bar{\chi}_1 := \chi_1 + \chi_{1,B}$.

In the other case that $b = 0$, for the query $(\mathbf{R}, x)$ of $\mathcal{A}$ to $\mathsf{SimSampMLWE\text{-}RU}^b$, the response $((\mathbf{A}_r, \mathbf{c}, \mathbf{h}), (\mathbf{B}_x, \mathbf{y}_B, \mathbf{h}_B))$ returned by the $\mathsf{SampHMLWEP}^b$ oracle has $\mathbf{y}_B$ uniformly random in R_q^h and $\mathbf{c}$ uniformly random and independent in R_q^N, but hints are still given by $\mathbf{h} = \mathbf{R}\widetilde{\mathbf{e}} + \widetilde{\mathbf{e}'}$ and $\mathbf{h}_B = \widetilde{\mathbf{e}_B} + \widetilde{\mathbf{e}'_B}$, and $\widetilde{\mathbf{e}'}$ (resp. $\widetilde{\mathbf{e}'_B}$) sampled from $-\chi_1$ (resp. $-\chi_{1,B}$), and $\widetilde{\mathbf{e}_B} + \widetilde{\mathbf{e}'_B}$ sampled from the distribution $\bar{\chi}_B := \chi_B + \chi'_B$ so it follows that $\mathbf{u} = \mathbf{R}\mathbf{c} + \mathbf{y}_B - (\mathbf{h} + \mathbf{h}_B + \mathbf{e}'_B) = \mathbf{R}\mathbf{c} + \mathbf{y}_B - (\mathbf{R}\widetilde{\mathbf{e}} + \widetilde{\mathbf{e}_B} + \mathbf{e}'_B) - (\widetilde{\mathbf{e}'} + \widetilde{\mathbf{e}'_B})$, where $-(\widetilde{\mathbf{e}'} + \widetilde{\mathbf{e}'_B})$ is sampled from the distribution $\chi_1 + \chi_{1,B}$, which is exactly the same as the distribution in the real $\mathsf{iMLWE\text{-}RU\text{-}G}^0$ game. We conclude that the advantage of $\mathcal{B}$ against HintMLWE-PRF is equal to the advantage of $\mathcal{A}$ against iMLWE-RU-G, as claimed in the case $\bar{\chi}_1 := \chi_1 + \chi_{1,B}$. The last part of the Theorem follows from the convolution Lemma 2 that implies that $\mathcal{D}_{\mathbb{Z}^{hd},s_1} + \mathcal{D}_{\mathbb{Z}^{hd},s_{1,B}}$ is within statistical distance 2ϵ of $\mathcal{D}_{\mathbb{Z}^{hd},\bar{s}_1}$ with $\bar{s}_1 := \sqrt{s_1^2 + s_{1,B}^2}$, and $\mathcal{D}_{\mathbb{Z}^{hd},s_B} + \mathcal{D}_{\mathbb{Z}^{hd},s'_B}$ is within statistical distance 2ϵ of $\mathcal{D}_{\mathbb{Z}^{hd},\bar{s}_B}$ with $\bar{s}_B := \sqrt{s_B^2 + (s'_B)^2}$ assuming the smooth-

```
$\mathcal{B}^{\mathsf{SampHMLWEP}^b(\cdot,\cdot),\mathsf{SampMLWE\text{-}PRF}^b(\cdot)}_{\mathcal{G},Q,Q_x^\infty,q,m,N,h,L,\beta_r,\bar{\chi}_h}(1^\lambda)$
$b' \leftarrow \mathcal{A}^{\mathsf{SimSampMLWE\text{-}RU}^b(\cdot,\cdot),\mathsf{SimSampMLWE\text{-}PRF}^b(\cdot)}(1^\lambda)$
$\textbf{return } b = b'$

$\mathsf{SimSampMLWE\text{-}PRF}^b(x \in \{0,1\}^L)$
$\textbf{if } \mathcal{Z}[x] = \perp \textbf{ then}$
    $\mathbf{y}'_B := \mathsf{SampMLWE\text{-}PRF}^b(x)$
    $s'_B := \sqrt{\bar{s}_B^2 - s_B^2}$
    $\mathbf{e}'_B \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^d, s'_B}$
    $\mathbf{y}_B = \mathbf{y}'_B + \mathbf{e}'_B$
    $\mathcal{Z}[x] := \mathbf{y}_B$
    $\mathcal{E}[x] := \mathbf{e}'_B$
$\textbf{return } \mathcal{Z}[x]$

$\mathsf{SimSampMLWE\text{-}RU}^b(\mathbf{R} \in R^{h \times N}, x \in \{0,1\}^L)$
$((\mathbf{A}_r, \mathbf{c}, \mathbf{h}), (\mathbf{B}_x, \mathbf{y}'_B, \mathbf{h}_B)) := \mathsf{SampHMLWEP}^b(\mathbf{R}, x)$
$\mathbf{e}'_B := \mathcal{E}[x]$
$\mathbf{u} = \mathbf{R}\mathbf{c} + \mathbf{y}'_B - (\mathbf{h} + \mathbf{h}_B)$
$\textbf{return } (\mathbf{A}_r, \mathbf{c}, \mathbf{u})$
```

Fig. 6: HintMLWE-PRF algorithm $\mathcal{B}$ in the security reduction.

ing condition $\min(\bar{s}_1, \bar{s}_B) \geq \sqrt{\ln(2hd(1 + 1/\epsilon))/\pi}$ is satisfied. So for Q queries the statistical distance is at most $4Q \cdot \epsilon$, as claimed. □

3.2 Hardness Reduction from MLWE-PRF to HintMLWE-PRF

Our reduction from MLWE-PRF to HintMLWE-PRF can be viewed as a special case of the reduction for the Matrix-Hint-MLWE problem in [ENP24] from the MLWE problem, which is a generalization of the reduction from MLWE to Hint-MLWE in [KLSS23]. However, unlike the Matrix-Hint-MLWE problem in [ENP24], which only involves MLWE samples with respect to *random* matrices, our HintMLWE-PRF problem also provides the adversary with MLWE samples with respect to *non-random* matrices $\mathbf{B}_x := \mathcal{G}(x)$ for the chosen instantiation of $\mathcal{G}$ (which is not necessarily a random oracle). That is the reason why we need to also provide the adversary with MLWE samples with respect to non-random matrices $\mathcal{G}(x)$ in the MLWE-PRF problem. Therefore, our reduction to HintMLWE-PRF is from MLWE-PRF, rather than a reduction from MLWE, as in [ENP24].

For completeness, we provide in the Appendix a complete proof of the reduction from MLWE-PRF to HintMLWE-PRF based on adaptation of the general result in [ENP24] to our Matrix Hint MLWE problem variant HintMLWE-PRF, which provides the adversary with oracles for both MLWE samples $(\mathbf{A}_{r,j}, \mathbf{c}_j = \mathbf{A}_{r,j}\mathbf{k} + \mathbf{e}_j)$ with a matrix hint $\mathbf{h}_j := \mathbf{R}_j\mathbf{e}_j + \mathbf{e}'_j$ about the error $\mathbf{e}_j$ with respect to matrix $\mathbf{R}_j$, as well as a hint $\mathbf{h}_{B,j} = \mathbf{e}_B + \mathbf{e}'_{B,j}$ about the error $\mathbf{e}_B$ in the PRF MLWE oracle sample $(\mathbf{B}_x, \mathbf{y}_B = \mathbf{B}_x\mathbf{k} + \mathbf{e}_B)$. These are given for $j \in [Q_x]$, where Q_x denotes the number of oracle queries on the same x values, which re-use the same PRF output, and hence the same $\mathbf{e}_B$ PRF error term.

We note that it may be possible to reduce from MLWE to MLWE-PRF with the instantiation of $\mathcal{G}$ as in [BP14], by a variant of the pseudorandomness reduction in [BP14] from MLWE. We plan to investigate this in future work.

Theorem 2. *Let $\bar{\chi}_h := (\chi, \chi_B, \chi_1, \chi_{1,B}, \chi_{k})$ be discrete distributions over R, where $\chi := \mathcal{D}_{\mathbb{Z}^d,s}$, $\chi_B := \mathcal{D}_{\mathbb{Z}^d,s_B}$, $\chi_1 := \mathcal{D}_{\mathbb{Z}^d,s_1}$ and $\chi_{1,B} := \mathcal{D}_{\mathbb{Z}^d,s_{1,B}}$ for some $s, s_B, s_1, s_{1,B}, \eta, \delta, \epsilon, \epsilon_p \in \mathbb{R}^+$. Let $\mathcal{G}$ be a hash family, and let $Q, Q_x^\infty, q, m, N, h, L, \beta_r \in \mathbb{Z}^+$. Moreover, let $\bar{\chi} = (\chi_0, \chi_0, \chi_{k})$, where $\chi_0 := \mathcal{D}_{\mathbb{Z}^d,s_0}$. The $\mathsf{HintMLWE\text{-}PRF}_{\mathcal{G},Q,Q_x^\infty,q,m,N,h,L,\beta_r,\bar{\chi}_h}$ assumption holds if the $\mathsf{MLWE\text{-}PRF}_{\mathcal{G},Q,Q,q,m,h,L,\bar{\chi}}$ assumption holds and the following are satisfied:*

$$s_1/s \geq \eta \cdot \beta_r d\sqrt{hN}, \quad s_{1,B}/s_B \geq \eta \cdot \sqrt{Q_x^\infty} \tag{2}$$

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

$$s_0 \geq \sqrt{\frac{(1+1/\delta) \cdot \ln(2\max(N,h)d(1+1/\epsilon))}{\pi}}, \tag{4}$$

$$\min(s, s_B) \geq \sqrt{\frac{\ln(2\max(N,h)d) + 4}{\pi\epsilon_p}}. \tag{5}$$

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

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{HintMLWE\text{-}PRF}}(\lambda) \leq \mathsf{Adv}_{\mathcal{B}}^{\mathsf{MLWE\text{-}PRF}}(\lambda) + 4Q \cdot \epsilon,$$

where Q and Q_x^∞ denote the total number of queries and maximum number of hints requested per x query, respectively, that the adversary $\mathcal{A}$ makes.

The proof of the above theorem follows 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 Section 1).

We are now ready to prove our security reduction from $\mathsf{MLWE\text{-}PRF}$ to $\mathsf{HintMLWE\text{-}PRF}$.

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

Note that for the discrete Gaussian sampling in line 13 to be ppt sampleable, it is sufficient by Lemma 1 that $\mathbf{\Sigma}_j$ is positive definite, i.e. $\sigma_{\min}(\mathbf{\Sigma}_j) > 0$, where $\mathbf{\Sigma}_j := \mathbf{\Sigma}_{0,j} - s_0^2\mathbf{I}$, and that[8] $\delta_p \cdot M_{\mathbf{\Sigma}_j} \leq 1$, where $\delta_p :=$

[8] 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 ϵ_p in the bound.

```
$\mathcal{B}^{\mathsf{SampMLWE\text{-}PRF}^b(\cdot),\mathsf{SampMLWE}^b(\cdot)}_{\mathcal{G},Q,Q^{\infty}_x,q,m,N,h,L,\beta_r,\bar{\chi}}(1^\lambda)$
$b' \leftarrow \mathcal{A}^{\mathsf{SimSampHMLWEP}^b(\cdot,\cdot),\mathsf{SimSampPRF}^b(\cdot)}(1^\lambda)$
**return** $b'$
```

```
$\mathsf{SimSampHMLWEP}^b(\mathbf{R} \in R^{h\times N}, x \in \{0,1\}^L)$
$j[x] := j[x] + 1, j := j[x], \mathbf{R}_j := \mathbf{R}$
**if** $\|\mathbf{R}_j\|_\infty > \beta_r$ **then return** $\perp$
$\mathbf{B}_x = \mathcal{G}(x) \in R_q^{h\times m}$
$\mathbf{y}_B := \mathsf{SampMLWE\text{-}PRF}^b(x)$
$\mathbf{h}_{B,j} := \mathbf{h}_B[x,j]$
$\widetilde{\mathbf{e}_B} \xleftarrow{\$} \chi_B^h$   // simulated error
Call $\mathsf{SampMLWE}^b(\cdot)$ $N$ times to get $(\mathbf{A}_{r,j}, \mathbf{c}'_j) \in R_q^{N\times m} \times R_q^N$
$\widetilde{\mathbf{e}'_j} \xleftarrow{\$} \chi_1^h$, $\widetilde{\mathbf{e}_j} \xleftarrow{\$} \chi^N$   // simulated errors
$\mathbf{h}_j = \mathbf{R}_j\widetilde{\mathbf{e}_j} + \widetilde{\mathbf{e}'_j}$
$\mathbf{\Sigma}_1 := s^2\mathbf{I} \in \mathbb{Z}^{Nd}$
$\mathbf{\Sigma}_{0,j} := \left(\mathbf{\Sigma}_1^{-1} + \frac{1}{s_1^2}\mathbf{R}_j^\top\mathbf{R}_j\right)^{-1}$
$\mathbf{d}_j := \frac{1}{s_1^2}\mathbf{\Sigma}_{0,j}\mathbf{R}_j^\top\mathbf{h}_j$
$\mathbf{t}_j \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^{Nd},\mathbf{\Sigma}_{0,j}-s_0^2\mathbf{I},\mathbf{d}_j}$
$\mathbf{c}_j := \mathbf{c}'_j + \mathbf{t}_j \in R_q^N$
**return** $((\mathbf{A}_{r,j}, \mathbf{c}_j, \mathbf{h}_j), (\mathbf{B}_x, \mathbf{y}_B, \mathbf{h}_{B,j}))$
```

```
$\mathsf{SimSampPRF}^b(x)$
    $\mathbf{y}_B := \mathcal{Z}_{sim}[x]$
**if** $\mathcal{Z}_{sim}[x] = \perp$ **then**
    $\widetilde{\mathbf{e}_B} \xleftarrow{\$} \chi_B^h$
    **for** $i \in [Q_x^\infty]$ **do**
        $\mathbf{e}'_{B,i} \xleftarrow{\$} \chi_{1,B}^h$
        $\mathbf{h}_B[x,j] := \widetilde{\mathbf{e}_B} + \mathbf{e}'_{B,i}$
    **endfor**
    $\mathbf{R}_B^\top := [\mathbf{I} \cdots \mathbf{I}] \in \mathbb{Z}^{hd \times Q_x^\infty hd}$
    $\mathbf{h}_B^\top := [\mathbf{h}_{B,1}^\top, \ldots, \mathbf{h}_{B,Q_x^\infty}^\top]$
    $\mathbf{\Sigma}_{1,B} := s_B^2\mathbf{I} \in \mathbb{Z}^{hd\times hd}$
    $\mathbf{\Sigma}_{0,B} := \left(\mathbf{\Sigma}_{1,B}^{-1} + \frac{1}{s_{1,B}^2}\mathbf{R}_B^\top\mathbf{R}_B\right)^{-1}$
    $\mathbf{d}_B := \frac{1}{s_{1,B}^2}\mathbf{\Sigma}_{0,B}\mathbf{R}_B^\top\mathbf{h}_B$
    $\mathbf{t}_B \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^{hd},\mathbf{\Sigma}_{0,B}-s_0^2\mathbf{I},\mathbf{d}_B}$
    $\mathbf{y}'_B := \mathsf{SampMLWE\text{-}PRF}^b(x)$
    $\mathbf{y}_B := \mathbf{y}'_B + \mathbf{t}_B \in R_q^h, \mathcal{Z}_{sim}[x] := \mathbf{y}_B$
**return** $\mathbf{y}_B$
```

Fig. 7: MLWE-PRF algorithm $\mathcal{B}$ in the security reduction.

$\sqrt{\frac{\ln(2Nd)+4}{\pi}}$ and $M_{\mathbf{\Sigma}_j}$ denotes the max column norm of $\sqrt{\mathbf{\Sigma}_j^{-1}}$, which is bounded as $M_{\mathbf{\Sigma}_j} \leq \sqrt{\sigma_{\max}(\mathbf{\Sigma}_j^{-1})} \leq \frac{1}{\sqrt{\sigma_{\min}(\mathbf{\Sigma}_j)}} = \frac{1}{\sqrt{\sigma_{\min}(\mathbf{\Sigma}_{0,j}) - s_0}^2}$. So the sufficient sampleability condition is

$$\sigma_{\min}(\mathbf{\Sigma}_{0,j}) \geq (1+\delta)\cdot s_0^2 + \delta_p^2, \tag{6}$$

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

$$\begin{aligned}
\|\mathbf{\Sigma}_{0,j}^{-1}\| &= \|\mathbf{\Sigma}_1^{-1} + \frac{1}{s_1^2}\mathbf{R}^\top\mathbf{R}\| \\
&\leq \|\mathbf{\Sigma}_{1,j}^{-1}\| + \frac{1}{s_1^2}\|\mathbf{R}_j^\top\mathbf{R}_j\| \\
&\leq \frac{1}{s^2} + \frac{1}{s_1^2}\|\mathbf{R}_j\|^2 \qquad (7)\\
&\leq \frac{1}{s^2} + \frac{1}{s_1^2}((\beta_r d)^2 hN) \\
&\leq \frac{1}{s^2}\cdot(1 + 1/\eta^2).
\end{aligned}$$

In the first inequality above we use the triangle inequality. The second inequality uses the identity $\|\mathbf{R}^\top \mathbf{R}\| = \|\mathbf{R}\|^2$. The third inequality uses the upper bound $\|\mathbf{R}_j\| \leq \beta_r d\sqrt{Nh}$ using the assumed bound β_r on the infinity norm of the entries of $\mathbf{R}_j$, and the fourth inequality uses the assumed lower bound (2) on s_1/s. Therefore, using (7), we see that the sampleability condition (6) is implied by the assumed condition (2) on s_1/s, the condition (3) and the condition $\epsilon_p \geq \delta_p^2/s$, which is implied by condition (5), as required.

Similarly, for the discrete Gaussian sampling in line 13 of $\mathsf{SimSampPRF}$ 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 $\mathbf{\Sigma}_B := \mathbf{\Sigma}_{0,B} - s_0^2\mathbf{I}$ and $\delta_{pB} \cdot M_{\mathbf{\Sigma}_j} \leq 1$, where $\delta_{pB} := \sqrt{\frac{\ln(2hd)+4}{\pi}} \leq \delta_{pB}$ and $M_{\mathbf{\Sigma}_B}$ denotes the max column norm of $\sqrt{\mathbf{\Sigma}_B^{-1}}$, which is bounded as $M_{\mathbf{\Sigma}_B} \leq \sqrt{\sigma_{\max}(\mathbf{\Sigma}_B^{-1})} \leq \frac{1}{\sqrt{\sigma_{\min}(\mathbf{\Sigma}_B)}} = \frac{1}{\sqrt{\sigma_{\min}(\mathbf{\Sigma}_{0,j})-s_0}^{2}}$. So the sufficient sampleability condition is

$$\sigma_{\min}(\mathbf{\Sigma}_{0,B}) \geq (1+\delta) \cdot s_0^2 + \delta_{pB}^2, \tag{8}$$

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

$$\begin{aligned} \|\mathbf{\Sigma}_{0,B}^{-1}\| &= \left\| \mathbf{\Sigma}_{1,B}^{-1} + \frac{1}{s_{1,B}^2}\mathbf{R}_B^\top \mathbf{R}_B \right\| \\ &\leq \|\mathbf{\Sigma}_{1,B}^{-1}\| + \frac{1}{s_{1,B}^2}\|\mathbf{R}_B^\top \mathbf{R}_B\| \\ &\leq \frac{1}{s_B^2} + \frac{Q_x^\infty}{s_{1,B}^2} \\ &\leq \frac{1}{s_B^2} \cdot (1 + 1/\eta^2). \end{aligned} \tag{9}$$

In the first inequality above we use the triangle inequality. The second inequality uses the identity $\|\mathbf{R}_B^\top \mathbf{R}_B\| = \|\mathbf{R}_B\|^2$ The third inequality uses the fact $\mathbf{R}_B^\top \mathbf{R}_B = Q_x^\infty \mathbf{I}$, and the fourth inequality uses the assumed lower bound (2) on $s_{1,B}/s$. Therefore, using (9), we see that the sampleability condition (8) is implied by the assumed condition (2) on $s_{1,B}/s_B$, the condition (3) and the condition $\epsilon_{pB} \geq \delta_{pB}^2/s_B$, which is implied by condition (5), as required.

Next, to analyze the advantage of $\mathcal{B}$ against $\mathsf{MLWE\text{-}PRF}$, we consider first the case that $b = 1$. In this case, consider a query $(\mathbf{R}_j, x)$ by $\mathcal{A}$ to $\mathsf{SimSampHMLWEP}^b$. The vector $\mathbf{c}'_j$ returned by the $\mathsf{SampMLWE}^1$ has the form

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

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

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

where $\mathbf{e}_j := \mathbf{e}'_j + \mathbf{t}_j$ is sampled from the distribution $\mathcal{D}_{\text{conv}} := \mathcal{D}_{\mathbb{Z}^{Nd}, s_0^2\mathbf{I}} + \mathcal{D}_{\mathbb{Z}^{Nd}, \mathbf{\Sigma}_{0,j} - s_0^2\mathbf{I}, \mathbf{d}_j}$. By the convolution Lemma 2, the distribution $\mathcal{D}_{\text{conv}}$ 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:

$$\sqrt{\mathbf{\Sigma}_3} \geq \eta_\epsilon(\mathbb{Z}^{Nd}), \text{ where } \mathbf{\Sigma_3} := \frac{1}{s_0^2}\mathbf{I} + \left(\mathbf{\Sigma}_{0,j} - s_0^2\mathbf{I}\right)^{-1}. \tag{12}$$

We show below that (12) is satisfied. Therefore, by the one matrix hint conditional distribution Lemma 5, the first triple $((\mathbf{A}_{r,j}, \mathbf{c}_j, \mathbf{h}_j)_{j \in [Q_x]})$ returned by the $\mathsf{SimSampHMLWEP}^1$ simulator for each queried x is within statistical distance $\leq 2\epsilon$ of the distribution of the output returned by the $\mathsf{SampHMLWEP}^1$ oracle in the real $\mathsf{HintMLWE\text{-}PRF}$ game. Note that the j'th response hint $\mathbf{h}_j$ provides a hint about the j'th secret errors $\mathbf{e}_j$, 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 $j \in [Q_x^\infty]$. Adding up over all $\leq Q$ x queries, we conclude that the joint distribution of the tuples $((\mathbf{A}_{r,j}, \mathbf{c}_j, \mathbf{h}_j)_{j \in [Q_x]})$ in the view of $\mathcal{A}$ simulated by $\mathcal{B}$ in the $\mathsf{MLWE\text{-}PRF}$ game with $b = 1$ is within statistical distance $\leq 2Q\epsilon$ of the distribution of $\mathcal{A}$'s view in the real $\mathsf{HintMLWE\text{-}PRF}$ game with $b = 1$.

Similarly, the PRF output $\mathbf{e}'_B$ returned by the $\mathsf{SampMLWE\text{-}PRF}^1$ oracle for a query x has the form

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

where $\mathbf{B}_x = \mathcal{G}(x)$ and $\mathbf{e}'_B$ is sampled from the distribution $\mathcal{D}_{\mathbb{Z}^{hd}, s_B^2 \mathbf{I}}$ and hence

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

where $\mathbf{e}_B := \mathbf{e}'_B + \mathbf{t}_B$ is sampled from the distribution $\mathcal{D}_{\text{convB}} := \mathcal{D}_{\mathbb{Z}^{hd}, s_0^2\mathbf{I}} + \mathcal{D}_{\mathbb{Z}^{hd}, \mathbf{\Sigma}_{0,B} - s_0^2\mathbf{I}, \mathbf{d}_B}$. By the convolution Lemma 2, the distribution $\mathcal{D}_{\text{convB}}$ 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:

$$\sqrt{\mathbf{\Sigma}_{3,B}} \geq \eta_\epsilon(\mathbb{Z}^{hd}), \text{ where } \mathbf{\Sigma}_{3,B} := \frac{1}{s_0^2}\mathbf{I} + \left(\mathbf{\Sigma}_{0,B} - s_0^2\mathbf{I}\right)^{-1}. \tag{15}$$

We show below that (15) is satisfied. Therefore, by the matrix hint conditional distribution Lemma 5 with respect to the Q_x^∞-hint matrix $\mathbf{R}_B := [\mathbf{I} \cdots \mathbf{I}]^\top \in \mathbb{Z}^{Q_x^\infty hd \times hd}$ (since all Q_x^∞ 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 second triples $(\mathbf{B}_x, \mathbf{y}_B, \mathbf{h}_{B,j})_{j \in [Q_x]}$ returned by the $\mathsf{SimSampHMLWEP}^1$ simulator for each queried x is within statistical distance $\leq 2\epsilon$ of the distribution of the output returned by the $\mathsf{SampHMLWEP}^1$ oracle in the real $\mathsf{HintMLWE\text{-}PRF}$ game. Adding up over 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 $\mathsf{MLWE\text{-}PRF}$ game with $b = 1$ is within statistical distance $\leq 2Q\epsilon$ of the distribution of $\mathcal{A}$'s view in the real $\mathsf{HintMLWE\text{-}PRF}$ game with $b = 1$. So the overall view of $\mathcal{A}$ simulated by $\mathcal{B}$ in the $\mathsf{MLWE\text{-}PRF}$ game with $b = 1$ is within statistical distance $\leq 4Q\epsilon$ of the distribution of $\mathcal{A}$'s view in the real $\mathsf{HintMLWE\text{-}PRF}$ game with $b = 1$.

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 $\mathbb{Z}_q^{hd}$ and $\mathbb{Z}_q^{hd}$ respectively, and 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 $\mathbb{Z}_q^{hd}$ and $\mathbb{Z}_q^{hd}$ respectively, independently of the hint vectors $\mathbf{h}_j$ and $\mathbf{h}_{B,j}$, while the marginal distribution of the hint vectors $\mathbf{h}$ remains the same as in the $b = 1$ game. Therefore, the distribution of the view of $\mathcal{B}$ simulated by $\mathcal{A}$ in the $\mathsf{MLWE\text{-}PRF}$ game with $b = 0$ is exactly equal to the distribution of $\mathcal{B}$'s view in the real $\mathsf{HintMLWE\text{-}PRF}$ game with $b = 0$. From the above analysis, we conclude that $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{MLWE\text{-}PRF}}(\lambda) \geq \mathsf{Adv}_{\mathcal{A}}^{\mathsf{HintMLWE\text{-}PRF}}(\lambda) - 4Q \cdot \epsilon$, which is the claimed advantage bound.

It remains to show that conditions (12) and (15) are satisfied. By Lemma 4, the condition (12) holds if

$$\|\mathbf{\Sigma}_3^{-1}\| \leq \eta_\epsilon(\mathbb{Z}^{Nd})^{-2}. \tag{16}$$

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

$$\|\mathbf{\Sigma}_3^{-1}\| \leq \frac{1}{s_0^2} + \|(\mathbf{\Sigma}_{0,j} - s_0^2\mathbf{I})^{-1}\| = \frac{1}{s_0^2} + \frac{1}{\sigma_{\min}(\mathbf{\Sigma}_{0,j} - s_0^2\mathbf{I})} \leq \frac{1 + 1/\delta}{s_0^2}. \tag{17}$$

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

$$\|\mathbf{\Sigma}_{3,B}^{-1}\| \leq \frac{1}{s_0^2} + \|(\mathbf{\Sigma}_{0,B} - s_0^2\mathbf{I})^{-1}\| = \frac{1}{s_0^2} + \frac{1}{\sigma_{\min}(\mathbf{\Sigma}_{0,B} - s_0^2\mathbf{I})} \leq \frac{1 + 1/\delta}{s_0^2}. \tag{19}$$

Combining (19) and (18), we conclude that (15) is satisfied if $s_0 \geq \sqrt{1 + 1/\delta} \cdot \eta_\epsilon(\mathbb{Z}^{hd})$, and the latter condition is implied by the assumed lower bound (4) on s_0, using Lemma 3, as required. □

4 LeOPaRd : Our Lattice-Based VPOPRF Proposal

At a very high-level, our goal is to realise the 2HashDH OPRF idea [JKK14] in the lattice setting as first done by Albrecht et al. [ADDS21]. Our construction ensures that a client with a tag/input pair (t, x) can recover the PRF value $\lfloor \mathbf{B}_x \mathbf{k} \rceil_p$ after interacting with the server, where $\mathbf{B}_x = \mathcal{G}(t, x)$, for some *message mapping* $\mathcal{G}$, and $\mathbf{k}$ is the secret key of the server. We discuss potential instantiations of the function $\mathcal{G}$ and its relation to the existing lattice-based PRF schemes in Section 5.2. As discussed before, our OPRF construction, LeOPaRd, is round-optimal, and supports partial obliviousness and verifiability.

We would like the OPRF evaluation to begin with the client linearly encrypting the matrix $\mathbf{B}_x$ such that $\mathbf{C}_x := \mathbf{R}\mathbf{A}_r + \mathbf{B}_x$, 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 $\mathbf{B}_x \mathbf{k}$). However, to minimize the client's control over $\mathbf{C}_x$ so that $\mathbf{u}_x$ does not leak server's secret key, we ask the client to commit to the pair $(\mathbf{R}, x)$, 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 $\mathbf{R}$ and the input x *before* seeing the random matrix $\mathbf{A}_r$. This step is critical to ensuring that we can reduce pseudorandomness of LeOPaRd to the new iMLWE-RU-R problem (without requiring smudging). The client sends the pair $(\mathbf{c}_r, \mathbf{C}_x)$ (and their well-formedness NIZK proof) to the server.

Upon receiving the values from the client, the server recomputes $\mathbf{A}_r$ given the commitment $\mathbf{c}_r$, and responds to the client with the pair $(\mathbf{u}_x := \mathbf{C}_x \mathbf{k} + \mathbf{e}'_s, \mathbf{v}_k := \mathbf{A}_r \mathbf{k} + \mathbf{e}_s)$, where $\mathbf{e}_s$ and $\mathbf{e}'_s$ denote some appropriately distributed errors. At this point, the client can use its secret randomness $\mathbf{R}$ to recover the PRF output $\lfloor \mathbf{u}_x - \mathbf{R}\mathbf{v}_k \rceil_p = \lfloor \mathbf{B}_x \mathbf{k} + \mathbf{e}_f \rceil_p = \lfloor \mathbf{B}_x \mathbf{k} \rceil_p$, where $\mathbf{e}_f := \mathbf{e}'_s - \mathbf{R}\mathbf{e}_s$ 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 Figure 8, and the NIZK relations of client and server proofs (π_c, π_s) are given in Figure 9. In the protocol description, NIZK_1 and 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 in LeOPaRd using Regev-style encryption [Reg05], which can also be seen as an extractable version of the BDLOP commitment [BDL+18]. Given such an encryption is quite standard by now, we defer the details to Appendix E.

In our LeOPaRd description in Figure 8, 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 π_s. There are important applications of 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 1 (Threshold VPOPRF). Although we presented (for the sake of simplicity) our construction from Figure 8 in a single server setting, we note that it easily extends to n-out-of-n setting, as 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 blind signing, each server computes the tuple $(\mathbf{u}_{x,i}, \mathbf{v}_{k,i}, \pi_{s,i})$ identically as in lines 8-9 and 12 of F.BlindEval in Figure 8. Finally, the user replaces line 6 of F.Finalize from Figure 8 with the computation $\mathbf{z} := \lfloor \sum_{i=1}^{n} \mathbf{u}_{x,i} - \mathbf{R} \sum_{i=1}^{n} \mathbf{v}_{k,i} \rceil_p$ in order to obtain the final OPRF value.

Remark 2. 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 E, 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 3. In our iMLWE-RU-G security reduction from Section 3, it is important to know the maximum number of OPRF evaluation queries the adversary makes with the *same* $\mathbf{B}_x = \mathcal{G}(t, x)$. In the partially

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

```
$\forall i \in [2], \mathsf{ck}_i \leftarrow \mathsf{COM.Setup}(1^\lambda)$
$\forall i \in [2], \mathsf{crs}_i \leftarrow \mathsf{NIZK}_i.\mathsf{Setup}(1^\lambda)$
$\mathsf{pp} := (\mathsf{ck}_1, \mathsf{ck}_2, \mathsf{crs}_1, \mathsf{crs}_2)$
$\mathbf{return}\ \mathsf{pp}$
```

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

```
$\mathbf{parse}\ \mathsf{pp} := (\mathsf{ck}_1, \mathsf{ck}_2, \mathsf{crs}_1, \mathsf{crs}_2)$
$\mathbf{k} \xleftarrow{\$} \chi_k^m$
$\mathbf{c}_k \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_2, \mathbf{k}; \rho_k)$
$\mathbf{return}\ (\mathsf{pk} := \mathbf{c}_k, \mathsf{sk} := \mathbf{k})$
```

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

```
$\mathbf{parse}\ \mathsf{pp} := (\mathsf{ck}_1, \mathsf{ck}_2, \mathsf{crs}_1, \mathsf{crs}_2)$
$\mathbf{R} \xleftarrow{\$} \chi_r^{h \times (\ell+m)}$
$\mathbf{c}_r \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_1, (\mathbf{R}, x); \rho_r)$
$\mathbf{A}_r := \mathsf{RO}_r(t, \mathbf{c}_r) \in R_q^{(\ell+m) \times m}$
$\mathbf{B}_x := \mathcal{G}(t, x) \in R_q^{h \times m}$
$\mathbf{C}_x := \mathbf{R}\mathbf{A}_r + \mathbf{B}_x \in R_q^{h \times m}$
$\mathsf{stmt}_1 := (\mathbf{c}_r, \mathbf{C}_x, \mathsf{ck}_1, \mathbf{A}_r, \mathcal{G}, t)$
$\mathsf{wit}_1 := (\mathbf{R}, x, \rho_r)$
$\pi_c \leftarrow \mathsf{NIZK}_1.\mathsf{P}(\mathsf{crs}_1, \mathsf{stmt}_1, \mathsf{wit}_1)$
$\mathsf{st} := (t, x, \mathbf{R}, \mathbf{C}_x, \mathsf{pk}, \mathbf{A}_r)$
$\mathsf{req} := (\mathbf{c}_r, \mathbf{C}_x, \pi_c)$
$\mathbf{return}\ (\mathsf{st}, \mathsf{req})$
```

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

```
$\mathbf{parse}\ \mathsf{pp} := (\mathsf{ck}_1, \mathsf{ck}_2, \mathsf{crs}_1, \mathsf{crs}_2)$
$\mathbf{parse}\ \mathsf{pk} := \mathbf{c}_k, \mathsf{sk} := \mathbf{k}$
$\mathbf{parse}\ \mathsf{req} := (\mathbf{c}_r, \mathbf{C}_x, \pi_c)$
$\mathbf{A}_r := \mathsf{RO}_r(t, \mathbf{c}_r) \in R_q^{(\ell+m) \times m}$
$\mathbf{if}\ \mathsf{NIZK}_1.\mathsf{V}(\mathsf{crs}_1, \pi_c, \mathsf{stmt}_1) \neq 1\ \mathbf{then}$
    $\mathbf{return}\ \perp$
$\mathbf{e}_s \xleftarrow{\$} \chi^{\ell+m}, \mathbf{e}'_s \xleftarrow{\$} \chi_1^h$
$\mathbf{v}_k := \mathbf{A}_r\mathbf{k} + \mathbf{e}_s \in R_q^{\ell+m}$
$\mathbf{u}_x := \mathbf{C}_x\mathbf{k} + \mathbf{e}'_s \in R_q^h$
$\mathsf{stmt}_2 := (\mathbf{u}_x, \mathbf{C}_x, \mathbf{c}_k, \mathbf{v}_k, \mathsf{ck}_2, \mathbf{A}_r)$
$\mathsf{wit}_2 := (\mathbf{k}, \mathbf{e}_s, \mathbf{e}'_s, \rho_k)$
$\pi_s \leftarrow \mathsf{NIZK}_2.\mathsf{P}(\mathsf{crs}_2, \mathsf{stmt}_2, \mathsf{wit}_2)$
$\mathbf{return}\ \mathsf{rep} := (\mathbf{u}_x, \mathbf{v}_k, \pi_s)$
```

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

```
$\mathbf{parse}\ \mathsf{pp} := (\mathsf{ck}_1, \mathsf{ck}_2, \mathsf{crs}_1, \mathsf{crs}_2)$
$\mathbf{parse}\ \mathsf{rep} := (\mathbf{u}_x, \mathbf{v}_k, \pi_s)$
$\mathbf{parse}\ \mathsf{st} := (t, x, \mathbf{R}, \mathbf{C}_x, \mathsf{pk} := \mathbf{c}_k, \mathbf{A}_r)$
$\mathbf{if}\ \mathsf{NIZK}_2.\mathsf{V}(\mathsf{crs}_2, \pi_s, \mathsf{stmt}_2) \neq 1\ \mathbf{then}$
    $\mathbf{return}\ \perp$
$\mathbf{z} := \lfloor \mathbf{u}_x - \mathbf{R}\mathbf{v}_k \rceil_p \in R_p^h$    // $= \lfloor \mathbf{B}_x\mathbf{k} \rceil_p$
$y := \mathsf{RO}_z(t, x, \mathbf{z})$
$\mathbf{return}\ y$
```

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

```
$\mathbf{parse}\ \mathsf{sk} := \mathbf{k}$
$\mathbf{B}_x := \mathcal{G}(t, x) \in R_q^{h \times m}$
$\mathbf{z} := \lfloor \mathbf{B}_x\mathbf{k} \rceil_p \in R_p^h$
$y := \mathsf{RO}_z(t, x, \mathbf{z})$
$\mathbf{return}\ y$
```

$\mathsf{RO}_r(t, \mathbf{c}_r)$

```
$\mathbf{if}\ \mathcal{H}[t, \mathbf{c}_r] = \perp\ \mathbf{then}$
    $\mathcal{H}[t, \mathbf{c}_r] \xleftarrow{\$} R_q^{(\ell+m) \times m}$
$\mathbf{return}\ \mathcal{H}[t, \mathbf{c}_r]$
```

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

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

Fig. 8: LeOPaRd : Our verifiable POPRF construction.

The server proof π_s proves the following,

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

The client proof π_c proves the following,

$$\pi_c := \left\{ \begin{array}{c} \text{Ə}(\mathbf{c}_r, \mathbf{C}_x, \mathsf{ck}_1, \mathbf{A}_r, \mathcal{G}, t, \beta_r); \text{ꓘ}(\mathbf{R}, x, \rho_r) \mid \|\mathbf{R}\|_\infty \leq \beta_r \ \wedge \ \mathbf{B}_x := \mathcal{G}(t, x) \wedge \\ \mathbf{c}_r = \mathsf{COM.Commit}(\mathsf{ck}_1, (\mathbf{R}, x); \rho_r) \ \wedge \ \mathbf{C}_x = \mathbf{R}\mathbf{A}_r + \mathbf{B}_x \bmod q \end{array} \right\}.$$

Fig. 9: Relations for NIZK proofs performed in LeOPaRd.

Notation	**Description**
λ	security parameter
κ	correctness parameter
q	system modulus
p	rounding modulus
d	ring dimension of $R = \mathbb{Z}[X]/(X^d + 1)$
χ, χ_1	server's error distributions
β, β_1	ℓ_∞-norm bounds on server's errors $\mathbf{e}_s, \mathbf{e}'_s$
χ_k	server's key distribution
χ_r	client's randomness distribution
β_r	ℓ_∞-norm bound on client's randomness $\mathbf{R}$
s_0	error std. dev. in server's MLWE security
m	dimension of server key $\mathbf{k}$ (over R_q)
h	# of rows of $\mathbf{B}_x$
ℓ	client MLWE dimension parameter
n_c, n_s	dimensions of trapdoor keys used by client, server
γ	base parameter for Gadget matrix

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

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 $\mathbf{B}_x$. In this case, the impact of the term Q_x^∞ in Theorem 2 would effectively diminish.

4.1 Batched Queries

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. Suppose the client wants to get evaluations on $(x_1, \ldots, x_K) =: \mathbf{x}$ with tags $(t_1, \ldots, t_K) =: \mathbf{t}$ for some $K \geq 1$. Then, we can run Request and BlindEval procedures as shown in Figure 10. Finalize would compute $\hat{\mathbf{z}} := \left\lfloor \hat{\mathbf{u}}_x - \hat{\mathbf{R}}\mathbf{v}_k \right\rceil_p$ and then output $y_i := \mathsf{RO}_z(x_i, t_i, \mathbf{z}_i)$ for $\hat{\mathbf{z}} = (\mathbf{z}_1, \ldots, \mathbf{z}_K)$ and $i = 1, \ldots, K$.

The batched LeOPaRd has two major advantages over running K independent evaluation queries:

1. Both the client and the server 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-based proof size increases from 45KB (for one single query) to just 79KB for a batch of 64 queries ($< 1.8\times$ increase).
2. Observe that the term $\mathbf{v}_k$ (on the server's side) in our protocol is *not* affected by batching at all. Therefore, its size remains constant for any $K \geq 1$. When using the message mapping from BP14 PRF (see Section 5.2), we have $h = 1$ and therefore, $\mathbf{t}_x$ is a single ring element. The linear communication cost by the server in the batched setting then boils down K ring elements.

```
F.Request(pp, pk, t, x)
parse $\mathbf{t} := (t_1, \ldots, t_K)$
parse $\mathbf{x} := (x_1, \ldots, x_K)$
parse $\mathsf{pp} := (\mathsf{ck}_1, \mathsf{ck}_2, \mathsf{crs}_1, \mathsf{crs}_2)$
$\hat{\mathbf{R}} \xleftarrow{\$} \chi_r^{Kh\times(\ell+m)}$
$\hat{\mathbf{c}}_r \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_1, (\hat{\mathbf{R}}, \mathbf{x}); \rho_r)$
$\mathbf{A}_r := \mathsf{RO}_r(\mathbf{t}, \hat{\mathbf{c}}_r) \in R_q^{(\ell+m)\times m}$
$\forall i \in [K],\ \mathbf{B}_{x_i} := \mathcal{G}(x_i, t_i) \in R_q^{h\times m}$
$\hat{\mathbf{B}}_x := \begin{pmatrix}\mathbf{B}_{x_1}\\ \vdots \\ \mathbf{B}_{x_K}\end{pmatrix}$
$\hat{\mathbf{C}}_x := \hat{\mathbf{R}}\mathbf{A}_r + \hat{\mathbf{B}}_x \in R_q^{Kh\times m}$
$\mathsf{stmt}_1 := (\hat{\mathbf{c}}_r, \hat{\mathbf{C}}_x, \mathsf{ck}_1, \mathbf{A}_r, \mathcal{G}, \mathbf{t})$
$\mathsf{wit}_1 := (\hat{\mathbf{R}}, \mathbf{x}, \rho_r)$
$\pi_c \leftarrow \mathsf{NIZK}_1.\mathsf{P}(\mathsf{crs}_1, \mathsf{stmt}_1, \mathsf{wit}_1)$
$\mathsf{st} := (\mathbf{x}, \hat{\mathbf{R}}, \hat{\mathbf{C}}_x, \mathsf{pk}, \mathbf{A}_r)$
$\mathsf{req} := (\hat{\mathbf{c}}_r, \hat{\mathbf{C}}_x, \pi_c)$
return $(\mathsf{st}, \mathsf{req})$
```

```
F.BlindEval(pp, pk, sk, t, req)
parse $\mathsf{pp} := (\mathsf{ck}_1, \mathsf{ck}_2, \mathsf{crs}_1, \mathsf{crs}_2)$
parse $\mathsf{pk} := \mathbf{c}_k, \mathsf{sk} := \mathbf{k}$
parse $\mathsf{req} := (\hat{\mathbf{c}}_r, \hat{\mathbf{C}}_x, \pi_c)$
$\mathbf{A}_r := \mathsf{RO}_r(\mathbf{t}, \hat{\mathbf{c}}_r) \in R_q^{(\ell+m)\times m}$
if $\mathsf{NIZK}_1.\mathsf{V}(\mathsf{crs}_1, \pi_c, \mathsf{stmt}_1) \neq 1$ then
    return $\perp$
$\mathbf{e}_s \xleftarrow{\$} \chi^{\ell+m}, \hat{\mathbf{e}}'_s \xleftarrow{\$} \chi_1^{Kh}$
$\mathbf{v}_k := \mathbf{A}_r\mathbf{k} + \mathbf{e}_s \in R_q^{\ell+m}$
$\hat{\mathbf{u}}_x := \hat{\mathbf{C}}_x\mathbf{k} + \hat{\mathbf{e}}'_s \in R_q^{Kh}$
$\mathsf{stmt}_2 := (\hat{\mathbf{u}}_x, \hat{\mathbf{C}}_x, \mathbf{c}_k, \mathbf{v}_k, \mathsf{ck}_2, \mathbf{A}_r)$
$\mathsf{wit}_2 := (\mathbf{k}, \mathbf{e}_s, \hat{\mathbf{e}}'_s, \rho_k)$
$\pi_s \leftarrow \mathsf{NIZK}_2.\mathsf{P}(\mathsf{crs}_2, \mathsf{stmt}_2, \mathsf{wit}_2)$
return $\mathsf{rep} := (\hat{\mathbf{u}}_x, \mathbf{v}_k, \pi_s)$
```

Fig. 10: Batched LeOPaRd evaluation protocol.

The significant practical advantage in the batched setting can be observed from the results presented in Table 3 and Table 4.

4.2 Correctness and Security Analyses

Correctness analysis. We upper bound the correctness error probability of our protocol in Lemma 6, in terms of the protocol parameters. Due to space limitations, we only sketch the proofs and refer the reader to Appendix B.2 for full proofs and details.

Lemma 6. *Fix κ, 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 (30)) that holds except with probability $p_e \leq 2^{-(\kappa+2+\log(hd))}$. Also, assume that the function family $\mathcal{G}$ satisfies ϵ_u-*uniformity *(as per Definition 19) with*

$$\epsilon_u \leq 2^{-(\kappa+2+\log(hd))}. \tag{20}$$

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

$$q/p \geq 2^{\kappa+2} \cdot hd \cdot (2B_f(\kappa, d) + 1). \tag{21}$$

Remarks:

- Assume that $\chi = \mathcal{U}(\mathbb{S}_\beta)$, $\chi_r = \mathcal{U}(\mathbb{S}_{\beta_r})$ 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 (30), 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 β_1 for the coordinates of $\mathbf{e}'_s$, we have the approximate upper bound

$$B_f(\kappa, d) \leq \beta_1 + \sigma\sigma_r\sqrt{(m+\ell)d \cdot 2\ln(2)(\kappa + 2 + \log(hd))}. \tag{22}$$

Note that $B_f(\kappa, d)$ is also upper bounded by the worst-case bound ($p_e = 0$):

$$B_f(\kappa, d) \leq (m+\ell)d\beta\beta_r + \beta_1 \tag{23}$$

- 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_{eval} total key generation and protocol evaluation pairs, the bound in (21) will be multiplied by Q_{eval}.

In our evaluation protocol, the client obtains $\mathbf{u}_x - \mathbf{R}\mathbf{v}_k = \mathbf{B}_x\mathbf{k} + \mathbf{e}_f \in R_q^h$, 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 $\Pr[\lfloor \mathbf{B}_x\mathbf{k} + \mathbf{e}_f \rceil_p \neq \lfloor \mathbf{B}_x\mathbf{k} \rceil_p]$, which occurs only if $\mathbf{B}_x\mathbf{k}$ falls "close" to (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 ϵ_u, of coordinates of $\mathbf{B}_x\mathbf{k}$ (a property of $\mathcal{G}$ which we call ϵ_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 ϵ_u-uniformity of the BLMR13 and BP14 PRF instantiations of $\mathcal{G}$ for this purpose.

Security analysis. We first prove pseudorandomness against malicious clients with Theorem 3 and then request privacy against malicious servers (POPRIV2) with Theorem 4. Due to space constraints, we give only the sketch of the main proof steps, and refer the reader to Appendix B.3 and Appendix B.4 for the full proofs, respectively. In Appendix C.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 3. *The POPRF construction* F *from Figure 8 satisfies pseudorandomness given in Definition 3, with random oracles* RO_r *and* RO_z*, if:*

- *The client argument system* NIZK_1 *is computationally sound and the server argument system* NIZK_2 *is computationally zero-knowledge (Definition 17),*
- *The commitment scheme* COM *is computationally hiding and extractable (Definitions 14 and 16), and*
- *The* $\mathsf{iMLWE\text{-}RU\text{-}R}_{\mathsf{param}}$ *assumption, for* $\mathsf{param} := (\mathcal{G}, Q_{\mathsf{M}}, Q_{\mathsf{P}}, Q_x^\infty, q, m, \ell + m, h, 2\lambda, \beta_r, p, \bar{\chi})$*, holds (Definition 18).*

More precisely, for any PPT *adversary* $\mathcal{A}$*, there exist* PPT *adversaries* $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$*,* $\mathcal{B}_4$ *and* $\mathcal{B}_5$ *against the computational zero-knowledge of* NIZK_2*, computational soundness of* NIZK_1*, hiding of* COM*, extractability of* COM *and* $\mathsf{iMLWE\text{-}RU\text{-}R}_{\mathsf{param}}$ *assumption, respectively, such that*

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

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

Our proof essentially revolves around the idea of removing traces of the server's secret key sk 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 π_s, such that from that point onwards we do not need sk for the generation of the proof. Next, we commit to an all zero vector $\mathbf{0}_m$ during the computation of the public key pk, which allows us to remove sk from the computation of pk. Lastly, we rely on our $\mathsf{iMLWE\text{-}RU\text{-}R}$ assumption to remove all traces of sk from the computation of $\mathbf{u}_x$ and $\mathbf{v}_k$, and yet maintain the correctness of the scheme. After applying these changes, we end up in a position where we do not require the secret key sk anymore, and hence, the simulator $\mathcal{S}$ can be constructed easily. We note that to make these argument go through, we also need to extract the client's randomness $\mathbf{R}$ and carefully program the random oracles RO_r and RO_z throughout the proof.

Theorem 4. *The POPRF construction* F *from Figure 8 satisfies the request privacy against malicious servers (POPRIV2), given in Definition 4, with random oracles* RO_r *and* RO_z*, if:*

- *The client argument system* NIZK_1 *is computationally zero-knowledge and the server argument system* NIZK_2 *is computationally sound (Definition 17),*
- *The commitment scheme* COM *is computationally hiding and extractable (Definitions 14 and 16), 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* $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$*,* $\mathcal{B}_4$ *and* $\mathcal{B}_5$ *against the computational zero-knowledge of* NIZK_1*, computational hiding of* COM*, computational soundness of* NIZK_2*, extractability of* COM *and* $\mathsf{knMLWE}_{m,\ell+m,h,\chi}$ *assumption, respectively, such that*

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

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

In order to prove request privacy against malicious servers, we aim to remove traces of the 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 π_c, such that we do not require anymore the input x for computing the proof. Next, we change the client commitment $\mathbf{c}_r$ to a commitment of all zero vector $\mathbf{0}_{h\cdot(\ell+m)+1}$ instead of the pair $(\mathbf{R}, x)$, which removes another occurrence of the input x. Finally, we rely on the Knapsack MLWE (knMLWE) 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 sk from the public key pk, and use sk to correct the POPRF evaluation in $\mathsf{Finalize}$. Importantly, we can use Regev-style [Reg05] *lightweight* commitments, as opposed to requiring *heavy* full trapdoors to extract from a commitment of the form $\mathbf{A}\mathbf{k} + \mathbf{e}$ used in [ADDS21]. At this point we removed all occurrences of the input x, and hence, the adversary's view is independent of the input bit b.

5 Instantiating the NIZK Proofs and Message Mapping $\mathcal{G}$

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

5.1 Proof by the Server

The NIZK proof, π_s, (see Figure 9) 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 E, 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 6.

5.2 Instantiating the Message Mapping $\mathcal{G}$

In this section, we primarily focus on the *secret-dependent* part of the message mapping $\mathcal{G}$ and discuss our main option on how the message mapping $\mathcal{G} : x \mapsto \mathbf{B}_x \in R_q^{h \times m}$ 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 $\mathcal{G}(\hat{x})$ and it is easy to verify the computation w.r.t. this input extended with *public* information (which we discuss more in Section 5.3). More message mapping options, including a random oracle, are discussed in Appendix D. There are two main considerations: (i) security of $\mathsf{iMLWE\text{-}RU\text{-}R}$ w.r.t. to $\mathcal{G}$ 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 $\mathsf{iMLWE\text{-}RU\text{-}R}$ remains secure for all instantiations discussed in this work.

We believe that a natural way to instantiate $\mathcal{G}$ is to use the mappings in existing lattice-based PRFs so that the $\mathsf{Samp\text{-}PRF}$ oracle in $\mathsf{iMLWE\text{-}RU\text{-}R}$ becomes an oracle outputting 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 $\mathcal{G}$. The option we see as the most suitable one is using the message mapping employed in BP14 PRF [BP14]. Here, the PRF computation involves a gadget matrix $\mathbf{G}$ and its (non-linear) inverse computation $G^{-1} : R_q^{1 \times m} \to R_q^{m \times m}$. The G^{-1} mapping is the standard bit decomposition operation. In this case, we have two vectors $\mathbf{a}_0, \mathbf{a}_1 \xleftarrow{\$} R_q^m$ published as public parameters, and as a result, $h = 1$. Then, for $\mathbf{k} \leftarrow \chi_k^m$ for some distribution χ_k[9] on R_q, we compute

$$F_{\mathbf{k}}^{\mathsf{bp}}(x) = \left\lfloor \mathbf{b}_x^\top \cdot \mathbf{k} \right\rceil_p , \tag{24}$$

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

$$\mathcal{G}^{\mathsf{bp}} : x \mapsto \mathbf{b}_x^\top = \mathbf{a}_{x_0}^\top \cdot G^{-1}(\mathbf{a}_{x_1} \cdots G^{-1}(\mathbf{a}_{x_{L-2}}^\top \cdot G^{-1}(\mathbf{a}_{x_{L-1}}))). \tag{25}$$

For the computation to correctly work, we need $m = \lceil \log q \rceil$. For the client's NIZK proof in this case, we will need to treat each $\mathbf{B}_i := G^{-1}(\mathbf{a}_{x_i}^\top \cdot \mathbf{B}_{i+1}) \in R_q^{m \times m}$ for $i = L-1, \ldots, 0$ (with $\mathbf{B}_L := \mathbf{I}_m$) as a variable. Therefore, there are $m^2 L$ variable polynomials, but with the correctness restriction that $m = \lceil \log q \rceil$.

We 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 require that $m = \lceil \log_\gamma q \rceil$, and hence, a smaller m parameter may suffice.

5.3 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 9) 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 $\mathbf{B}_x = \mathcal{G}(t, x)$. Accordingly, we next discuss the proof of $\mathbf{B}_x = \mathcal{G}(t, x)$. Depending on the instantiation of the message mapping $\mathcal{G}$, the client's proof may vary significantly. We look more closely at the first two instantiations of $\mathcal{G}$ discussed in Section 5.2.

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

NIZK for BP14 PRF mapping. Recall we have $h = 1$. We first discuss the case when no tag t is present, i.e., $\mathbf{B}_x = \mathcal{G}(x)$. The case of $\mathbf{B}_x = \mathcal{G}(t, x)$ 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 $x = (x_0, x_1, \dots, x_{L-1}) \in \{0,1\}^L$ such that $\mathbf{C}_x = \mathbf{R}\mathbf{A}_r + \mathbf{B}_x \bmod q$ where $\mathbf{B}_x$ is defined in (25).

Using a similar strategy as in [ADDS21] (see "**Proof system 1: Proofs of Masked Partial PRF Computation**"), define variables $\mathbf{B}_i \in R_q^{m \times m}$ for $i = L-1, \dots, 0$ as $\mathbf{B}_{L-1} := G^{-1}(\mathbf{a}_{x_{L-1}}^\top)$ and $\mathbf{B}_i := G^{-1}(\mathbf{a}_{x_i}^\top \cdot \mathbf{B}_{i+1})$ for $i = L-2, \dots, 0$. Using this, we have $\mathbf{B}_x = \mathbf{G} \cdot \mathbf{B}_0$. Then the statement being proven now can be transformed to prove the following system with $\mathbf{B}_L := \mathbf{I}$:

$$\begin{aligned} \mathbf{G} \cdot \mathbf{B}_i &= \mathbf{a}_{x_i} \cdot \mathbf{B}_{i+1}, \quad i = 0, \dots, L-1 \\ \mathbf{C}_x &= \mathbf{R}\mathbf{A}_r + \mathbf{G} \cdot \mathbf{B}_0. \end{aligned}$$

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

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

The required proof now boils down to proving that a sequence of quadratic equations holds, which can be easily handled by LaBRADOR [BS23]. Observe that the witness dimension (over R_q) is in the order of $m^2 L$. As discussed in Section 5.2, we consider a generalized version of the BP14 PRF with the gadget matrix with base $\gamma \geq 2$. Therefore, the witness dimension (over R_q) is in the order of $m^2 L \approx (\log_\gamma q)^2 L$. In our parameter settings (see Section 6), the largest m is around 64; hence $m^2 L \approx 2^{18}$ (or less) for $L = 64$. 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 $\mathbf{B}_x = \mathcal{G}(t, x)$ (i.e., $\mathbf{B}_x = \mathcal{G}(\hat{x})$ for $\hat{x} := x \| t$), 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.

6 Practical Performance Analysis

Our performance analysis focuses on estimating the sizes of various components of 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 LeOPaRd is the size of the modulus q. Since the dimension parameters for suitable 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 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 6), the size of q itself is heavily dominated by the 2^κ term. Therefore, given this bound is *statistical* and for correctness (rather than a security property), we consider a range of κ 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 5, our correctness analysis easily extends to cover such a malicious server setting as the NIZK proof π_s proves shortness of the server's error terms, and also we can simply increase κ. 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. Therefore, 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 $\mathcal{G}^{\mathsf{bp}}$ with the BP14 PRF since this choice leads to a better performance overall. As discussed in Section 5, we consider a general base γ for the gadget matrix $\mathbf{G}$, i.e., $\mathbf{g} = (1, \gamma, \gamma^2, \ldots)$, and $\mathbf{G} = \mathbf{g} \otimes \mathbf{I}$. As a result, we only require $m \geq \lceil \log_\gamma q \rceil$[10]. We first consider the size of q (i.e., $\log q$), then determine m based on the other requirements such as MLWE security and then set $\gamma = \lceil 2^{\log q/m} \rceil$.

For simplicity (as in the correctness analysis), we set $\chi = \mathcal{U}(\mathbb{S}_\beta)$, $\chi_r = \mathcal{U}(\mathbb{S}_{\beta_r})$ and $\chi_1 = \mathcal{U}(\mathbb{S}_{\beta_1})$ for parameters β, β_1 and β_r[11]. Since the client's randomness $\mathbf{R}$ is one-time, we simply fix $\beta_r = 1$. Note that β_r does not impact the $\mathsf{iMLWE\text{-}RU\text{-}R}$ security reduction in Section 3. Since BP14 PRF allows for a small secret key, we sample the server's key $\mathbf{k}$ from the same χ distribution, i.e., $\chi_{_k} = \chi = \mathcal{U}(\mathbb{S}_\beta)$.

To estimate the practical hardness of $\mathsf{iMLWE\text{-}RU\text{-}R}$ problem, we rely on the reduction in Section 3, particularly Theorem 2. Based on this theorem, we require the hardness of MLWE with the same secret dimension m as in $\mathsf{iMLWE\text{-}RU\text{-}R}$ and error Gaussian parameter s_0. We also require the size of the rounding errors to be greater than the s parameter given in Theorem 2, which is easily satisfied since q/p is very large in our case.

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 3). 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. Afterwards, we set $\bar{s}_B$ of Theorem 1 to $\frac{q}{\sqrt{12p}}\sqrt{2\pi}$ so that the corresponding std. dev. is roughly $\frac{q}{\sqrt{12p}}$, matching the rounding error. Then, based on Theorem 2, we first fix $s_B = s$, $\delta = \eta = 1$, $Q = 2^{64}$ and $\epsilon = 2^{-128}$, and compute the smallest $(s_0, s, s_1, s_{1,B})$ values. The parameter $\bar{s}_1 = \sqrt{s_{1,B}^2 + s_1^2}$ is set as in Theorem 1. We then convert $(s_0, s, \bar{s}_1)$ to standard deviations $(\sigma_0, \sigma, \sigma_1)$ (by dividing by $\sqrt{2\pi}$). 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. We then find the corresponding smallest β (and β_1) such that uniform distribution on $\{-\beta, \ldots, \beta\}$ (and $\{-\beta_1, \ldots, \beta_1\}$) has standard deviation at least σ (and σ_1). In this procedure, we have $N = m + \ell$ and we estimate the values for m and ℓ initially, and correct the estimate iteratively until the estimate is checked to be accurate.

Now we consider the size of q based on the correctness analysis (Lemma 6) and set $\log q$ as small as possible. Then, we set the smallest value for m so that MLWE security with error standard deviation σ_0 at 128 bits is achieved (server security). As discussed above, this comes from the security reduction in Section 3. Similarly, we set the smallest value for ℓ 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

[10] Note that if $m > \lceil \log_\gamma q \rceil$, then we can simply pad the output vectors of G^{-1} decomposition mapping with zeros.

[11] It is common in practice to use uniform distributions while the security reduction between lattice problems (as our Theorem 2) goes through using Gaussian distributions. For example, the NIST-standardized Dilithium signature [DKL+18] uses no Gaussian distribution. We use uniform distributions for simplicity here and our reported performance results would not be significantly altered by switching to Gaussian.

that given many MLWE and iMLWE-RU-R 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. Given that we are dealing with (relatively) large moduli, we see that the third requirement 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 E used by the server and client, respectively). As discussed in Appendix E, we require the same MLWE assumptions over R_q with errors with standard deviation σ_0 (for server's commitment) and errors sampled from χ_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.

To estimate the sizes of LaBRADOR [BS23] (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 under a 75-bit modulus for FHE ciphertexts is about 45 KB for a single query and 79 KB for a batch of 64 queries. We believe the proofs needed for LeOPaRd are even simpler than those in [ADDG24], and therefore, we take our proof cost to be 45 KB for a single query and 79 KB for a batch of 64 queries when estimating sizes for LeOPaRd. We note that the dimension of the polynomial ring used in the core LeOPaRd parts and that in the underlying NIZK proofs do not necessarily need to be the same as discussed in [ADDG24, LNPS21]. However, our results turn out to be optimal for $d = 64$, which is the dimension already used in LaBRADOR.

In Table 3, we present some example parameter settings and communication sizes aiming at 128-bit security level. The offline communication simply involves communication of the server's public key, while online communication is those by the client and server. Here, we consider two cases: (i) fully oblivious setting where $Q_x^\infty = 2^\kappa$ (for Q_x^∞ in Theorem 2); and (ii) partially oblivious setting where $Q_x^\infty = 2^{16}$. In the former case, we are restricting the total number of OPRF queries to 2^κ, 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.

Acknowledgements. 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, and Australian Research Council Discovery Grants DP180102199 and 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.

Obliv. mode	Query batch size	κ	$m = n_s$	$\ell = n_c$	σ_0	β	β_1	$\log q$	Server commun. (per query)	Server PK size	Client commun. (per query)
Partial	1	16	24	27	3.95	15	7782	42	62.06	13.5	76.27
		32	33	37	3.95	15	8838	58	77.17	26.81	105.42
		64	53	56	3.96	15	10691	91	123.20	70.39	195.45
	64	16	24	27	3.95	15	7782	42	1.82	13.50	26.27
		32	33	37	3.95	15	8838	58	2.18	26.81	48.57
		64	53	56	3.96	15	10691	91	3.16	70.39	117.66
Full	1	16	24	27	3.95	15	7782	42	62.06	13.50	76.27
		32	37	41	4.09	16	1064838	65	85.12	34.11	120.88
		64	66	69	4.36	20	$\approx 2^{36}$	113	165.06	110.34	277.77
	64	16	24	27	3.95	15	7782	42	1.82	13.50	26.27
		32	37	41	4.09	16	1064838	65	2.36	34.11	60.41
		64	66	69	4.36	20	$\approx 2^{36}$	113	3.98	110.34	180.41

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 $d = 64$, $p = 8$, $h = 1$, $\gamma = 4$, and $\beta_r = 1$.

Obliv. mode	Query batch size	κ	$m = n_s$	$\ell = n_c$	σ_0	β	β_1	$\log q$	Server commun. (per query)	Server PK size	Client commun. (per query)
Partial	1	16	18	20	3.94	15	6971	42	57.80	10.12	68.39
		32	25	27	3.95	15	7841	58	69.02	20.31	90.05
		64	39	41	3.95	15	9347	90	101.95	51.19	154.36
	64	16	18	20	3.94	15	6971	42	1.76	10.12	20.01
		32	25	27	3.95	15	7841	58	2.06	20.31	36.73
		64	39	41	3.95	15	9347	90	2.82	51.19	86.00
Full	1	16	18	20	3.94	15	6971	42	57.80	10.12	68.39
		32	28	30	4.09	16	1062331	65	74.96	25.81	101.60
		64	49	52	4.36	20	$\approx 2^{36}$	113	135.05	81.92	219.34
	64	16	18	20	3.94	15	6971	42	1.76	10.12	20.01
		32	28	30	4.09	16	1062331	65	2.20	25.81	45.61
		64	49	52	4.36	20	$\approx 2^{36}$	113	3.51	81.92	135.18

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 $d = 64$, $p = 8$, $h = 1$, $\gamma \in \{5, 6\}$, and $\beta_r = 1$.

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. 1, 2, 3, 7, 8, 22, 29, 54

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. 1, 2, 3, 4, 5, 6, 20, 25, 27, 54, 55

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. 1, 2, 3, 4, 6, 20, 29

ALS20. Thomas Attema, Vadim Lyubashevsky, and Gregor Seiler. Practical product proofs for lattice commitments. In *CRYPTO (2)*, LNCS, pages 470–499. Springer, 2020. 25

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. 28, 29

BDFH25. Ward Beullens, Lucas Dodgson, Sebastian Faller, and Julia Hesse. The 2hash oprf framework and efficient post-quantum instantiations. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology – EUROCRYPT 2025*, pages 332–362, Cham, 2025. Springer Nature Switzerland. 1, 2, 3

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. 20, 59

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. 35

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, 11, 57, 58

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. 58

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. 10

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. 55

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, 11, 15, 26

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, 22, 25, 26, 27, 29, 58

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

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. 57, 58

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, 22

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. 34

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. 5, 28, 29

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. 10

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. 29

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, 12, 15, 16

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. 25

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). 27, 28

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). 25, 28, 29

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). 59

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). 28

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). 10, 27

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. 1

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. 3

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. 58

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. 59

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, *Advances in Cryptology – EUROCRYPT 2025*, pages 254–283, Cham, 2025. Springer Nature Switzerland. 3, 4

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, 3, 4, 20

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, 20

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, 20

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. 1

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, 54

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, 10, 12, 13, 15, 16

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, 20, 25, 28, 29, 59

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. 29, 59

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. 25, 27

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. 10

MS25. Daniele Micciancio and Adam Suhl. Simulation-secure threshold PKE from LWE with polynomial modulus. *IACR Communications in Cryptology*, 1(4), 2025. 6

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. 27

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. 10

Reg05. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93. ACM, 2005. 20, 25, 58

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. 37, 57

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, 7, 8, 9, 20, 24, 54, 55

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. 54

YBH$^+$25. Yibin Yang, Fabrice Benhamouda, Shai Halevi, Hugo Krawczyk, and Tal Rabin. Gold OPRF: Post-Quantum Oblivious Power-Residue PRF . In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 259–278, Los Alamitos, CA, USA, May 2025. IEEE Computer Society. 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. 58

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 13 (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* λ*, outputs a commitment key* ck*.*

$\mathsf{Commit}(\mathsf{ck}, m)$**:** *is a* PPT *algorithm that on input a commitment key* 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* 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 Commit definition, we leave the randomness part implicit and assume that it is generated internally inside the 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 *perfect binding* properties, along with *extractability.*

Definition 14 (Computational Hiding). *A commitment scheme* COM *is computationally hiding if for every* PPT *adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\lambda)$*, 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*

```
$\mathsf{CH}_{\mathsf{COM},\mathcal{A}}(\lambda)$
$\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda)$
$(m_0, m_1) \leftarrow \mathcal{A}(\mathsf{ck})$
$b \overset{\$}{\leftarrow} \{0,1\}$
$(c, d) \leftarrow \mathsf{Commit}(\mathsf{ck}, m_b)$
$b' \leftarrow \mathcal{A}(c)$
$\textbf{return } b = b'$
```

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

```
$\mathsf{CB}_{\mathsf{COM},\mathcal{A}}(\lambda)$
$\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda)$
$(c, d, d', m, m') \leftarrow \mathcal{A}(\mathsf{ck})$
assert $m \neq m'$
return $\mathsf{Verify}(\mathsf{ck}, c, d, m) = 1 \wedge \mathsf{Verify}(\mathsf{ck}, c, d', m') = 1$
```

Definition 16 (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* $\mathsf{negl}(\lambda)$*, such that*

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

and

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

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 it as follows a non-interactive zero-knowledge (NIZK) argument system [BFM88] as follows.

Definition 17 (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 λ, 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 π.*

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

We require NIZK *to meet the following properties:*

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

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

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

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

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

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

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 Deferred Definitions and Proofs

B.1 Definition of iMLWE-RU-R

Definition 18 ($\mathsf{iMLWE\text{-}RU\text{-}R}_{\mathcal{G},Q_{\mathsf{M}},Q_{\mathsf{P}},Q_x^\infty,q,m,N,h,L,\beta_r,p,\bar{\chi}}$)**.** *Let $\bar{\chi} = (\chi, \chi_1, \chi_k)$ be a discrete distribution over R, $\mathcal{G}$ be a hash family, and $Q, Q_x^\infty, q, m, N, h, L, \beta_r, p \in \mathbb{Z}^+$. We say that the interactive MLWE assumption* $\mathsf{iMLWE\text{-}RU\text{-}R}_{\mathsf{param}}$ *holds, for* $\mathsf{param} := (\mathcal{G}, Q_{\mathsf{M}}, Q_{\mathsf{P}}, Q_x^\infty, q, m, N, h, L, \beta_r, p, \bar{\chi})$*, if for all* PPT *adversaries $\mathcal{A}$, there exists a negligible function* $\mathsf{negl}(\cdot)$*, such that the following holds,*

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

where the experiment $\mathsf{iMLWE\text{-}RU\text{-}R}^b$ *is defined below, $Q_{\mathsf{M}}, Q_{\mathsf{P}}$ denote the total number of queries to* Samp-PRF*, and* SampMLWE-RU*, respectively, made by $\mathcal{A}$ during the experiment, and Q_x^∞ is the maximum allowed upper bound on the number of* SampMLWE-RU *queries* per queried *x.*

```
$\mathsf{iMLWE\text{-}RU\text{-}R}^b_{\mathcal{G},Q_{\mathsf{M}},Q_{\mathsf{P}},Q_x^\infty,q,m,N,h,L,\beta_r,p,\bar{\chi}}(\lambda)$
$\mathbf{k} \xleftarrow{\$} \chi_k^m$
$b' \leftarrow \mathcal{A}^{\mathsf{Samp\text{-}PRF}^b(\cdot),\mathsf{SampMLWE\text{-}RU}^b(\cdot,\cdot,\cdot)}(1^\lambda)$
return $b = b'$

$\mathsf{SampMLWE\text{-}RU}^b(\mathbf{R} \in R^{h \times N}, x \in \{0,1\}^L)$
if $\|\mathbf{R}\|_\infty > \beta_r$ then return $\perp$
$\mathbf{A}_r \xleftarrow{\$} R_q^{N \times m}, \mathbf{e} \xleftarrow{\$} \chi^N$
$\mathbf{B}_x = \mathcal{G}(x) \in R_q^{h \times m}$
$\mathbf{C}_x = \mathbf{R}\mathbf{A}_r + \mathbf{B}_x \in R_q^{h \times m}$, $\mathbf{e}' \xleftarrow{\$} \chi_1^h$
if $\mathcal{Z}[x] = \perp$ then $\mathsf{Samp\text{-}PRF}^0(x)$
$\mathbf{z}' := \mathcal{Z}[x]$
$\widetilde{\mathbf{e}}' \xleftarrow{\$} \chi_1^h$, $\widetilde{\mathbf{e}} \xleftarrow{\$} \chi^N$   // simulated errors
$\mathbf{e}_{\mathsf{sim}} = \widetilde{\mathbf{e}}' - \mathbf{R}\widetilde{\mathbf{e}} \in R^h$
if $b = 0$ then
    $\mathbf{c} \xleftarrow{\$} R_q^N$, $\mathbf{u} = \mathbf{R}\mathbf{c} + \mathbf{z}' + \mathbf{e}_{\mathsf{sim}} \in R_q^h$
if $b = 1$ then
    $\mathbf{c} = \mathbf{A}_r\mathbf{k} + \mathbf{e} \in R_q^N$, $\mathbf{u} = \mathbf{C}_x\mathbf{k} + \mathbf{e}' \in R_q^h$
return $(\mathbf{A}_r, \mathbf{c}, \mathbf{u})$

$\mathsf{Samp\text{-}PRF}^b(x)$
if $b = 0$ then
    if $\mathcal{Z}[x] = \perp$ then
        $\mathbf{z}' \xleftarrow{\$} R_q^h, \mathcal{Z}[x] := \mathbf{z}'$
    $\mathbf{z} = \lfloor \mathcal{Z}[x] \rceil_p$
if $b = 1$ then
    $\mathbf{B}_x = \mathcal{G}(x) \in R_q^{h \times m}$
    $\mathbf{z} = \lfloor \mathbf{B}_x\mathbf{k} \rceil_p \in R_p^h$
return $\mathbf{z}$
```

Note that the iMLWE-RU-R security game uses a table $\mathcal{Z}$ to store previous answers to $\mathsf{Samp\text{-}PRF}^0$ queries, to ensure that the same random answer is returned if the oracle is queried again at the same point x.

B.2 Correctness Analysis

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

$$\epsilon_u \leq 2^{-(\kappa+2+\log(hd))} \tag{26}$$

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

Lemma 6. *Fix κ, 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 (30)) that holds except with probability $p_e \leq 2^{-(\kappa+2+\log(hd))}$. Also, assume that the function family $\mathcal{G}$ satisfies ϵ_u-*uniformity *(as per Definition 19) with*

$$\epsilon_u \leq 2^{-(\kappa+2+\log(hd))}. \tag{20}$$

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

$$q/p \geq 2^{\kappa+2} \cdot hd \cdot (2B_f(\kappa, d) + 1). \tag{21}$$

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

$$\begin{align} \delta := \Pr[E] &:= \Pr[\lfloor \mathbf{u}_x - \mathbf{R}\mathbf{v}_k \rceil_p \neq \lfloor \mathbf{B}_x \mathbf{k} \rceil_p] \tag{27} \\ &= \Pr[\lfloor \mathbf{B}_x \mathbf{k} + \mathbf{e}_f \rceil_p \neq \lfloor \mathbf{B}_x \mathbf{k} \rceil_p] \tag{28} \\ & \tag{29} \end{align}$$

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

$$\mathbf{e}_f := \mathbf{u}_x - \mathbf{R}\mathbf{v}_k - \mathbf{B}_x \mathbf{k} = \mathbf{e}'_s - \mathbf{R}\mathbf{e}_s. \tag{30}$$

For $i \in [h]$ and $j \in [d]$, let $\mathbf{e}_{f,i} \in R$ denote the ith 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 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 $\mathbf{b}^\top_{x,i,j}\mathbf{k} \in \mathbb{Z}_q$ lands within distance $|\mathbf{e}_{f,i,j}|$ of a rounding interval boundary point in B_p. Therefore, we have

$$\begin{align} \delta &\leq \Pr[\exists i \in [h], j \in [d] \text{ s.t. } \mathbf{b}^\top_{x,i,j}\mathbf{k} \in B_p \pm |e_{f,i,j}|] \\ &\leq \sum_{i \in [h], j \in [d]} \delta_{i,j}, \text{ where } \delta_{i,j} := \Pr[\mathbf{b}^\top_{x,i,j}\mathbf{k} \in B_p \pm |e_{f,i,j}|]. \tag{31} \end{align}$$

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

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

where the first inequality above uses the assumed ϵ_u-uniformity of $\mathcal{G}$, 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 $\mathcal{G}$. For BLMR13 (see [SSS23] for the module instantiation), we have $h = m$, $\chi_k := U(R_q)$, $\mathcal{G}(\mathbf{x}) := \prod_{i=0}^{L-1} \mathbf{A}_{x_i}$, where $\mathsf{F.Setup}(1^\lambda)$ samples $\mathbf{A}_0, \mathbf{A}_1 \xleftarrow{\$} R_{\mathsf{bin}}^{m \times m}$ and restarts if $\mathbf{A}_0$ or $\mathbf{A}_1$ are not invertible over R_q.

Lemma 7. *For BLMR13, the function family $\mathcal{G}$ satisfies* 0*-uniformity. In particular, the distribution of* $\mathbf{B}_x\mathbf{k} := \mathcal{G}(x)\mathbf{k} \in R_q^{h\times m}$ *over the choice of* $\mathcal{G}$ *(in* $\mathsf{F.Setup}(1^\lambda)$*) and* $\mathbf{k} \xleftarrow{\$} \chi_k^m$ *is the uniform distribution on* $R_q^{h\times m}$*.*

Proof. Follows immediately from the uniformly random distribution χ_k and the invertibility of the matrix $\mathbf{B}_x = \mathcal{G}(x) = \prod_{i=0}^{L-1} \mathbf{A}_{x_i}$ over R_q, thanks to the invertibility of $\mathbf{A}_0$ and $\mathbf{A}_1$ over R_q. □

Bounds for BP14 PRF Instantiation of $\mathcal{G}$. For BP14 with $\chi_k := U(R_q)$, $\mathcal{G}(\mathbf{x}) := (\mathbf{A}_{x_0}\mathbf{B}_1')^\top \in R_q^{h\times m}$, where $\mathbf{B}_{L-1} := G^{-1}(\mathbf{A}_{x_{L-1}}) \in R_q^{m\ell\times m\ell}$, $\mathbf{B}_i := G^{-1}(\mathbf{A}_{x_i}\mathbf{B}_{i+1}) \in R_q^{m\ell\times m\ell}$ for $i = L-2,\dots,1$, $\mathbf{B}_1'$ consists of the leftmost h columns of $\mathbf{B}_1$, and $\mathsf{F.Setup}(1^\lambda)$ samples $\mathbf{A}_0, \mathbf{A}_1 \xleftarrow{\$} R_q^{m\times m\ell}$.

Lemma 8. *Assume that q is prime. If the BP14 function family $\mathcal{G}$ does not satisfy ϵ_u-uniformity for some non-negligible ϵ_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 ϵ between the distribution of $\mathbf{b}_{x,i,j}^\top\mathbf{k} \in \mathbb{Z}_q$ and $U(\mathbb{Z}_q)$ is $\geq \epsilon_u$, where $\mathbf{b}_{x,i,j}^\top$ 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 $\mathbf{B}_x \in R_q^{h\times m}$, and similarly $\mathbf{k}$ is viewed over $\mathbb{Z}$ as the concatanation of ring element coefficient vectors. Let Z denote the event that $\mathbf{b}_{x,i,j}^\top = \mathbf{0}$. Then we have $\epsilon = \epsilon_{nz}\cdot(1-\Pr[\mathsf{Z}]) + \epsilon_z\cdot\Pr[\mathsf{Z}]$, where ϵ_{nz} (resp. ϵ_z) denotes the statistical distance between the distribution of $\mathbf{b}_{i,j}^\top\mathbf{k} \in \mathbb{Z}_q$ and $U(\mathbb{Z}_q)$ conditioned on the event that Z does not occur (resp. Z occurs). Conditioned on Z not occuring, we have $\mathbf{b}_{x,i,j}^\top \neq \mathbf{0}$, and hence $\mathbf{b}_{x,i,j}^\top$ 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 $\mathbf{b}_{x,i,j}^\top\mathbf{k}$ 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 pseudornadomness adversary $\mathcal{A}$ against the BP14 PRF that works as follows: it queries x to its oracle to get $\mathbf{z} \in R_q^h$. If Z occurs then $\mathcal{A}$ outputs 1 if $z_{i,j} = 0$ and output 0 else. If 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 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 $\mathbb{Z}_p$, the event $z_{i,j} = 0$ will occur with probability $1/p$ independently of event 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 4. We note that the ϵ_u-uniformity Lemma above assumes χ_k has uniformly random coordinates in $\mathbb{Z}_q$, but the BP14 PRF could also be instantiated with χ_k restricted to small coefficients. In the latter case, the ϵ_u-uniformity seems more difficult to prove but we heuristically expect it to still be satisfied.

Remark 5. We note that our correctness analysis extends to a *malicious* server setting by relying on the fact that the NIZK proof π_s by the server ensures that the error terms $(\mathbf{e}_s, \mathbf{e}_s')$ have small coefficients (say bounded by β and β_1, respectively). Particularly, using the worst-case bound in (23) for B_f gives the correctness analysis requirement against a malicious server.

B.3 Pseudorandomness Analysis

Theorem 3. *The POPRF construction* F *from Figure 8 satisfies pseudorandomness given in Definition 3, with random oracles* RO_r *and* RO_z*, if:*

- *The client argument system* NIZK_1 *is computationally sound and the server argument system* NIZK_2 *is computationally zero-knowledge (Definition 17),*
- *The commitment scheme* COM *is computationally hiding and extractable (Definitions 14 and 16), and*
- *The* $\mathsf{iMLWE\text{-}RU\text{-}R_{param}}$ *assumption, for* $\mathsf{param} := (\mathcal{G}, Q_\mathsf{M}, Q_\mathsf{P}, Q_x^\infty, q, m, \ell+m, h, 2\lambda, \beta_r, p, \bar{\chi})$*, holds (Definition 18).*

More precisely, for any PPT *adversary* $\mathcal{A}$*, there exist* PPT *adversaries* $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$*,* $\mathcal{B}_4$ *and* $\mathcal{B}_5$ *against the computational zero-knowledge of* NIZK_2*, computational soundness of* NIZK_1*, hiding of* COM*, extractability of* COM *and* $\mathsf{iMLWE\text{-}RU\text{-}R_{param}}$ *assumption, respectively, such that*

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

where Q_{B} *and* Q_{RO_r} *denote the number of* $\mathsf{BlindEval}$ *and* RO_r *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 $\mathsf{Eval}, \mathsf{BlindEval}$ and Prim, where Prim denotes the random oracles RO_r and RO_z, and 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 $\mathbf{Hybrid_i}$ and $\mathbf{Hybrid_i} \approx \mathbf{Hybrid_{i+1}}$ denote $|\Pr[\mathbf{Hybrid_i} = 1] - \Pr[\mathbf{Hybrid_{i+1}} = 1]| \leq \mathsf{negl}(\lambda)$. We consider the hybrids listed below, and for each hybrid we depict the changes using figures. The figures include the whole protocol for ease of reference to remind how the real execution works, but we emphasize that we do not modify the adversarially controlled algorithms and parts.

$\mathbf{Hybrid_0}$: 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 Figure 8. Hence, it follows that

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

$\mathbf{Hybrid_1}$: In this hybrid, we replace crs_2 and proof π_s with a simulated CRS and proof $(\mathsf{crs}_2^*, \pi_s^*)$, as shown in Figure 11.

In order to show that $\mathbf{Hybrid_0} \approx \mathbf{Hybrid_1}$, we construct a reduction $\mathcal{B}_1$ to the computational zero-knowledge property of NIZK_2. We note that all queries are answered as in $\mathbf{Hybrid_0}$ with the exception of $\mathsf{BlindEval}$ queries. The only difference here is that $\mathcal{B}_1$ receives a CRS crs_2^* from its zero-knowledge challenger, and sets $\mathsf{crs}_2 := \mathsf{crs}_2^*$, instead of generating crs_2 via the $\mathsf{NIZK}_2.\mathsf{Setup}$ algorithm. When $\mathcal{A}$ makes a $\mathsf{BlindEval}$ query, $\mathcal{B}_1$ proceeds as in $\mathbf{Hybrid_0}$ to compute $\mathbf{v}_k$ and $\mathbf{u}_x$, but it makes an oracle call to its zero-knowledge challenger with the input $((\mathbf{u}_x, \mathbf{C}_x, \mathbf{c}_k, \mathbf{v}_k, \mathsf{ck}_2, \mathbf{A}_r), (\mathbf{k}, \mathbf{e}, \mathbf{e}_s, \mathbf{e}'_s))$ to obtain the proof π_s^*.

If the zero-knowledge challenger of NIZK_2 used the honest setup and prover algorithms to generate crs_2^* and π_s^*, 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 computational zero-knowledge property of NIZK_2. Hence, it follows that

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

and in particular $\mathbf{Hybrid_0} \approx \mathbf{Hybrid_1}$.

$\mathbf{Hybrid_2}$: Let BAD_1 be the event that for a blind evaluation query $(t, \mathsf{req} := (\mathbf{c}_r, \mathbf{C}_x, \pi_c))$, it holds that $\mathsf{NIZK}_1.\mathsf{V}(\mathsf{crs}_1, \pi_c, (\mathbf{c}_r, \mathbf{C}_x, \mathsf{ck}_1, \mathbf{A}_r, \mathcal{G}, t)) = 1$, but $\nexists(\mathbf{R}, x, \rho_r)$ such that $(\mathbf{C}_x = \mathbf{R}\mathbf{A}_r + \mathcal{G}(x) \ \wedge\ \mathbf{c}_r = \mathsf{COM.Commit}(\mathsf{ck}_1, (\mathbf{R}, x); \rho_r))$. We will prove that the BAD_1 event can only happen with negligible probability due to the soundness of NIZK_1 arguments system. Looking ahead, we need here such a reduction to the soundness of NIZK_1 because later in $\mathbf{Hybrid_4}$ we do a reduction to the extractability of the commitment scheme, and there we need to ensure that the adversarial inputs, specifically the commitment $\mathbf{c}_r$ is well-formed.

Clearly, we have that $\mathbf{Hybrid_1}$ and $\mathbf{Hybrid_2}$ are identical until the event 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

```
$\mathsf{F.Setup}(1^\lambda)$
$\forall i \in [2], \mathsf{ck}_i \leftarrow \mathsf{COM.Setup}(1^\lambda)$
$\mathsf{crs}_1 \leftarrow \mathsf{NIZK}_1.\mathsf{Setup}(1^\lambda)$
$\mathsf{crs}_2^*, \tau_2 \leftarrow \mathsf{NIZK}_2.\mathcal{S}_1(1^\lambda)$
$\mathsf{pp} := (\mathsf{ck}_1, \mathsf{ck}_2, \mathsf{crs}_1, \mathsf{crs}_2^*)$
$\textbf{return } \mathsf{pp}$
```

```
$\mathsf{F.KeyGen}(\mathsf{pp})$
$\textbf{parse } \mathsf{pp} := (\mathsf{ck}_1, \mathsf{ck}_2, \mathsf{crs}_1, \mathsf{crs}_2^*)$
$\mathbf{k} \xleftarrow{\$} \chi_k^m$
$\mathbf{c}_k \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_2, \mathbf{k}; \rho_k)$
$\textbf{return } (\mathsf{pk} := \mathbf{c}_k, \mathsf{sk} := \mathbf{k})$
```

```
$\mathsf{F.Request}(\mathsf{pp}, \mathsf{pk}, t, x)$
$\textbf{parse } \mathsf{pp} := (\mathsf{ck}_1, \mathsf{ck}_2, \mathsf{crs}_1, \mathsf{crs}_2^*)$
$\mathbf{R} \xleftarrow{\$} \chi_r^{h \times (\ell+m)}$
$\mathbf{c}_r \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_1, (\mathbf{R}, x); \rho_r)$
$\mathbf{A}_r := \mathsf{RO}_r(t, \mathbf{c}_r) \in R_q^{(\ell+m) \times m}$
$\mathbf{B}_x := \mathcal{G}(t, x) \in R_q^{h \times m}$
$\mathbf{C}_x := \mathbf{R}\mathbf{A}_r + \mathbf{B}_x \in R_q^{h \times m}$
$\mathsf{stmt}_1 := (\mathbf{c}_r, \mathbf{C}_x, \mathsf{ck}_1, \mathbf{A}_r, \mathcal{G}, t)$
$\mathsf{wit}_1 := (\mathbf{R}, x, \rho_r)$
$\pi_c \leftarrow \mathsf{NIZK}_1.\mathsf{P}(\mathsf{crs}_1, \mathsf{stmt}_1, \mathsf{wit}_1)$
$\mathsf{st} := (t, x, \mathbf{R}, \mathbf{C}_x, \mathsf{pk}, \mathbf{A}_r)$
$\mathsf{req} := (\mathbf{c}_r, \mathbf{C}_x, \pi_c)$
$\textbf{return } (\mathsf{st}, \mathsf{req})$
```

```
$\mathsf{F.BlindEval}(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}, t, \mathsf{req})$
$\textbf{parse } \mathsf{pp} := (\mathsf{ck}_1, \mathsf{ck}_2, \mathsf{crs}_1, \mathsf{crs}_2^*)$
$\textbf{parse } \mathsf{pk} := \mathbf{c}_k, \mathsf{sk} := \mathbf{k}$
$\textbf{parse } \mathsf{req} := (\mathbf{c}_r, \mathbf{C}_x, \pi_c)$
$\mathbf{A}_r := \mathsf{RO}_r(t, \mathbf{c}_r) \in R_q^{(\ell+m) \times m}$
$\textbf{if } \mathsf{NIZK}_1.\mathsf{V}(\mathsf{crs}_1, \pi_c, \mathsf{stmt}_1) \neq 1 \textbf{ then}$
    $\textbf{return } \perp$
$\mathbf{e}_s \xleftarrow{\$} \chi^{\ell+m}, \mathbf{e}'_s \xleftarrow{\$} \chi_1^h$
$\mathbf{v}_k := \mathbf{A}_r\mathbf{k} + \mathbf{e}_s \in R_q^{\ell+m}$
$\mathbf{u}_x := \mathbf{C}_x\mathbf{k} + \mathbf{e}'_s \in R_q^h$
$\mathsf{stmt}_2 := (\mathbf{u}_x, \mathbf{C}_x, \mathbf{c}_k, \mathbf{v}_k, \mathsf{ck}_2, \mathbf{A}_r)$
$\pi_s^* \leftarrow \mathsf{NIZK}_2.\mathcal{S}_2(\mathsf{crs}_2^*, \tau_2, \mathsf{stmt}_2)$
$\textbf{return } \mathsf{rep} := (\mathbf{u}_x, \mathbf{v}_k, \pi_s^*)$
```

```
$\mathsf{F.Finalize}(\mathsf{pp}, \mathsf{rep}, \mathsf{st})$
$\textbf{parse } \mathsf{pp} := (\mathsf{ck}_1, \mathsf{ck}_2, \mathsf{crs}_1, \mathsf{crs}_2^*)$
$\textbf{parse } \mathsf{rep} := (\mathbf{u}_x, \mathbf{v}_k, \pi_s^*)$
$\textbf{parse } \mathsf{st} := (t, x, \mathbf{R}, \mathbf{C}_x, \mathsf{pk} := \mathbf{c}_k, \mathbf{A}_r)$
$\textbf{if } \mathsf{NIZK}_2.\mathsf{V}(\mathsf{crs}_2^*, \pi_s^*, \mathsf{stmt}_2) \neq 1 \textbf{ then}$
    $\textbf{return } \perp$
$\mathbf{z} := \lfloor \mathbf{u}_x - \mathbf{R}\mathbf{v}_k \rceil_p \in R_p^h$    $/\!\!/ = \lfloor \mathbf{B}_x\mathbf{k} \rceil_p$
$y := \mathsf{RO}_z(t, x, \mathbf{z})$
$\textbf{return } y$
```

```
$\mathsf{F.Eval}(\mathsf{sk}, t, x)$
$\textbf{parse } \mathsf{sk} := \mathbf{k}$
$\mathbf{B}_x := \mathcal{G}(t, x) \in R_q^{h \times m}$
$\mathbf{z} := \lfloor \mathbf{B}_x\mathbf{k} \rceil_p \in R_p^h$
$y := \mathsf{RO}_z(t, x, \mathbf{z})$
$\textbf{return } y$
```

```
$\mathsf{RO}_r(t, \mathbf{c}_r)$
$\textbf{if } \mathcal{H}[t, \mathbf{c}_r] = \perp \textbf{ then}$
    $\mathcal{H}[t, \mathbf{c}_r] \xleftarrow{\$} R_q^{(\ell+m) \times m}$
$\textbf{return } \mathcal{H}[t, \mathbf{c}_r]$
```

```
$\mathsf{RO}_z(t, x, \mathbf{z})$
$\textbf{if } \mathcal{F}[t, x, \mathbf{z}] = \perp \textbf{ then}$
    $\mathcal{F}[t, x, \mathbf{z}] \xleftarrow{\$} \{0,1\}^\lambda$
$\textbf{return } \mathcal{F}[t, x, \mathbf{z}]$
```

Fig. 11: $\textbf{Hybrid}_1$ of pseudorandomness proof.

soundness of NIZK_1 argument system. We note that all queries are answered as in $\mathbf{Hybrid_1}$. Let Q_{B} denote the maximum number of $\mathsf{BlindEval}$ queries that $\mathcal{A}$ performs. $\mathcal{B}_2$ randomly chooses a $q^* \in [Q_{\mathsf{B}}]$ to serve as its guess for the query where the proof π_c will be forged by the adversary $\mathcal{A}$. For $\mathsf{req} \coloneqq (\mathbf{c}_r, \mathbf{C}_x, \pi_c)$ that corresponds to the q^*-th blind evaluation query, $\mathcal{B}_2$ simply outputs π_c. We emphasize that $\mathcal{B}_2$ avoids recovering $\mathbf{R}$, x or ρ_r (which are computationally infeasible to compute), and instead just guesses which of the proofs would have caused the event BAD_1 to happen. Hence, we have that

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

and in particular $\mathbf{Hybrid_1} \approx \mathbf{Hybrid_2}$.

$\mathbf{Hybrid_3}$: 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 the commitment $\mathbf{c}_k \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_2, \mathbf{k})$ as in the previous hybrid. This change is shown in Figure 12.

In order to show that $\mathbf{Hybrid_2} \approx \mathbf{Hybrid_3}$, we construct a reduction $\mathcal{B}_3$ to the computational hiding property of COM. For computing the public key 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 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 COM provided a commitment to m_0, then we are exactly in $\mathbf{Hybrid_2}$, and if it provided a commitment to m_1, then we are in $\mathbf{Hybrid_3}$. 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 COM. Hence, it follows that

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

and in particular $\mathbf{Hybrid_2} \approx \mathbf{Hybrid_3}$.

$\mathbf{Hybrid_4}$: In this hybrid, we replace the commitment key ck_1 with a trapdoor variant $(\mathsf{ck}^*_1, \mathsf{td}) \leftarrow \mathsf{COM}.\mathcal{E}_1(1^\lambda)$, and use the trapdoor td to extract the committed values queried to RO_r oracle, i.e., run $(\mathbf{R}, x) \coloneqq \mathsf{COM}.\mathcal{E}_2(\mathsf{ck}^*_1, \mathsf{td}, \mathbf{c}_r)$, compute $\mathbf{v}_k$ and $\mathbf{u}_x$ inside RO_r and store these values in a table for use during the blind evaluation queries, as depicted in Figure 13.

In order to prove that $\mathbf{Hybrid_3} \approx \mathbf{Hybrid_4}$ we construct an adversary $\mathcal{B}_4$ to the extractability property of COM. We note that all queries are answered as in $\mathbf{Hybrid_3}$ with the exception of $\mathsf{BlindEval}$ and RO_r queries. $\mathcal{B}_4$ samples a trapdoor commitment key $(\mathsf{ck}^*_1, \mathsf{td}) \leftarrow \mathsf{COM}.\mathcal{E}_1(1^\lambda)$, instead of generating ck_1 via the $\mathsf{COM.Setup}$ algorithm. Let Q_{RO_r} denote the number of RO_r queries that the adversary $\mathcal{A}$ makes. For each such query $(t, \mathbf{c}_r)$, $\mathcal{B}_4$ uses the trapdoor td to extract $(\mathbf{R}, x) \leftarrow \mathsf{COM}.\mathcal{E}_2(\mathsf{ck}^*_1, \mathsf{td}, \mathbf{c}_r)$ and compute the pair $(\mathbf{u}_x, \mathbf{v}_k)$ accordingly, which gets stored in the table $\mathcal{T}[t, \mathbf{c}_r] \coloneqq (\mathbf{u}_x, \mathbf{v}_k)$. Upon receiving a $\mathsf{BlindEval}$ query $(t, \mathsf{req} \coloneqq (\mathbf{c}_r, \mathbf{C}_x, \pi_c))$, $\mathcal{B}_4$ uses the already stored values in $\mathcal{T}[t, \mathbf{c}_r] \coloneqq (\mathbf{u}_x, \mathbf{v}_k)$ in order to simulate the proof π^*_s and answer the query with the tuple $(\mathbf{u}_x, \mathbf{v}_k, \pi^*_s)$.

Due to the soundness of the client proof π_c we know that the adversarial inputs, and especially commitments, are well-formed. Hence, if $\mathcal{A}$ can distinguish 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_{RO_r} queries to RO_r oracle, it follows that

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

and in particular $\mathbf{Hybrid_3} \approx \mathbf{Hybrid_4}$.

$\mathbf{Hybrid_5}$: In this hybrid, we change the way we respond to RO_r and Eval queries. Concretely, for answering Eval queries, instead of computing $\mathbf{z} \coloneqq \lfloor \mathbf{B}_x \mathbf{k} \rceil_p$, we sample a uniform $\mathbf{z}' \xleftarrow{\$} R_q^h$ and return $\mathbf{z} \coloneqq \lfloor \mathbf{z}' \rceil_p$. For answering RO_r queries, instead of computing $\mathbf{v}_k \coloneqq \mathbf{A}_r \mathbf{k} + \mathbf{e}_s$ and $\mathbf{u}_x \coloneqq \mathbf{C}_x \mathbf{k} + \mathbf{e}'_s$, we sample a uniform $\mathbf{v}_k \xleftarrow{\$} R_q^{\ell+m}$ and set $\mathbf{u}_x \coloneqq \mathbf{R}\mathbf{v}_k + \mathbf{z}' + \mathbf{e}^*$, where $\mathbf{e}^*$ is computed as $\mathbf{e}^* = \widetilde{\mathbf{e}}' - \mathbf{R}\widetilde{\mathbf{e}} \in R^h$, such that $\widetilde{\mathbf{e}}' \xleftarrow{\$} \chi_1^h, \widetilde{\mathbf{e}} \xleftarrow{\$} \chi^{\ell+m}$. These changes are depicted in Figure 14.

```
$\mathsf{F.Setup}(1^\lambda)$
$\forall i \in [2], \mathsf{ck}_i \leftarrow \mathsf{COM.Setup}(1^\lambda)$
$\mathsf{crs}_1 \leftarrow \mathsf{NIZK}_1.\mathsf{Setup}(1^\lambda)$
$\mathsf{crs}_2^*, \tau_2 \leftarrow \mathsf{NIZK}_2.\mathcal{S}_1(1^\lambda)$
$\mathsf{pp} \coloneqq (\mathsf{ck}_1, \mathsf{ck}_2, \mathsf{crs}_1, \mathsf{crs}_2^*)$
$\textbf{return } \mathsf{pp}$
```

```
$\mathsf{F.KeyGen}(\mathsf{pp})$
$\textbf{parse } \mathsf{pp} \coloneqq (\mathsf{ck}_1, \mathsf{ck}_2, \mathsf{crs}_1, \mathsf{crs}_2^*)$
$\mathbf{k} \xleftarrow{\$} \chi_k^m$
$\mathbf{c}_k^* \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_2, \mathbf{0}_m; \rho_k)$
$\textbf{return } (\mathsf{pk} \coloneqq \mathbf{c}_k^*, \mathsf{sk} \coloneqq \mathbf{k})$
```

```
$\mathsf{F.Request}(\mathsf{pp}, \mathsf{pk}, t, x)$
$\textbf{parse } \mathsf{pp} \coloneqq (\mathsf{ck}_1, \mathsf{ck}_2, \mathsf{crs}_1, \mathsf{crs}_2^*)$
$\mathbf{R} \xleftarrow{\$} \chi_r^{h \times (\ell+m)}$
$\mathbf{c}_r \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_1, (\mathbf{R}, x); \rho_r)$
$\mathbf{A}_r \coloneqq \mathsf{RO}_r(t, \mathbf{c}_r) \in R_q^{(\ell+m) \times m}$
$\mathbf{B}_x \coloneqq \mathcal{G}(t, x) \in R_q^{h \times m}$
$\mathbf{C}_x \coloneqq \mathbf{R}\mathbf{A}_r + \mathbf{B}_x \in R_q^{h \times m}$
$\mathsf{stmt}_1 \coloneqq (\mathbf{c}_r, \mathbf{C}_x, \mathsf{ck}_1, \mathbf{A}_r, \mathcal{G}, t)$
$\mathsf{wit}_1 \coloneqq (\mathbf{R}, x, \rho_r)$
$\pi_c \leftarrow \mathsf{NIZK}_1.\mathsf{P}(\mathsf{crs}_1, \mathsf{stmt}_1, \mathsf{wit}_1)$
$\mathsf{st} \coloneqq (t, x, \mathbf{R}, \mathbf{C}_x, \mathsf{pk}, \mathbf{A}_r)$
$\mathsf{req} \coloneqq (\mathbf{c}_r, \mathbf{C}_x, \pi_c)$
$\textbf{return } (\mathsf{st}, \mathsf{req})$
```

```
$\mathsf{F.BlindEval}(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}, t, \mathsf{req})$
$\textbf{parse } \mathsf{pp} \coloneqq (\mathsf{ck}_1, \mathsf{ck}_2, \mathsf{crs}_1, \mathsf{crs}_2^*)$
$\textbf{parse } \mathsf{pk} \coloneqq \mathbf{c}_k^*, \mathsf{sk} \coloneqq \mathbf{k}$
$\textbf{parse } \mathsf{req} \coloneqq (\mathbf{c}_r, \mathbf{C}_x, \pi_c)$
$\mathbf{A}_r \coloneqq \mathsf{RO}_r(t, \mathbf{c}_r) \in R_q^{(\ell+m) \times m}$
$\textbf{if } \mathsf{NIZK}_1.\mathsf{V}(\mathsf{crs}_1, \pi_c, \mathsf{stmt}_1) \neq 1 \textbf{ then}$
    $\textbf{return } \perp$
$\mathbf{e}_s \xleftarrow{\$} \chi^{\ell+m}, \mathbf{e}_s' \xleftarrow{\$} \chi_1^h$
$\mathbf{v}_k \coloneqq \mathbf{A}_r\mathbf{k} + \mathbf{e}_s \in R_q^{\ell+m}$
$\mathbf{u}_x \coloneqq \mathbf{C}_x\mathbf{k} + \mathbf{e}_s' \in R_q^h$
$\mathsf{stmt}_2 \coloneqq (\mathbf{u}_x, \mathbf{C}_x, \mathbf{c}_k^*, \mathbf{v}_k, \mathsf{ck}_2, \mathbf{A}_r)$
$\pi_s^* \leftarrow \mathsf{NIZK}_2.\mathcal{S}_2(\mathsf{crs}_2^*, \tau_2, \mathsf{stmt}_2)$
$\textbf{return } \mathsf{rep} \coloneqq (\mathbf{u}_x, \mathbf{v}_k, \pi_s^*)$
```

```
$\mathsf{F.Finalize}(\mathsf{pp}, \mathsf{rep}, \mathsf{st})$
$\textbf{parse } \mathsf{pp} \coloneqq (\mathsf{ck}_1, \mathsf{ck}_2, \mathsf{crs}_1, \mathsf{crs}_2^*)$
$\textbf{parse } \mathsf{rep} \coloneqq (\mathbf{u}_x, \mathbf{v}_k, \pi_s^*)$
$\textbf{parse } \mathsf{st} \coloneqq (t, x, \mathbf{R}, \mathbf{C}_x, \mathsf{pk} \coloneqq \mathbf{c}_k^*, \mathbf{A}_r)$
$\textbf{if } \mathsf{NIZK}_2.\mathsf{V}(\mathsf{crs}_2^*, \pi_s^*, \mathsf{stmt}_2) \neq 1 \textbf{ then}$
    $\textbf{return } \perp$
$\mathbf{z} \coloneqq \lfloor \mathbf{u}_x - \mathbf{R}\mathbf{v}_k \rceil_p \in R_p^h$    $/\!\!/ = \lfloor \mathbf{B}_x\mathbf{k} \rceil_p$
$y \coloneqq \mathsf{RO}_z(t, x, \mathbf{z})$
$\textbf{return } y$
```

```
$\mathsf{F.Eval}(\mathsf{sk}, t, x)$
$\textbf{parse } \mathsf{sk} \coloneqq \mathbf{k}$
$\mathbf{B}_x \coloneqq \mathcal{G}(t, x) \in R_q^{h \times m}$
$\mathbf{z} \coloneqq \lfloor \mathbf{B}_x\mathbf{k} \rceil_p \in R_p^h$
$y \coloneqq \mathsf{RO}_z(t, x, \mathbf{z})$
$\textbf{return } y$
```

```
$\mathsf{RO}_r(t, \mathbf{c}_r)$
$\textbf{if } \mathcal{H}[t, \mathbf{c}_r] = \perp \textbf{ then}$
    $\mathcal{H}[t, \mathbf{c}_r] \xleftarrow{\$} R_q^{(\ell+m) \times m}$
$\textbf{return } \mathcal{H}[t, \mathbf{c}_r]$
```

```
$\mathsf{RO}_z(t, x, \mathbf{z})$
$\textbf{if } \mathcal{F}[t, x, \mathbf{z}] = \perp \textbf{ then}$
    $\mathcal{F}[t, x, \mathbf{z}] \xleftarrow{\$} \{0,1\}^\lambda$
$\textbf{return } \mathcal{F}[t, x, \mathbf{z}]$
```

Fig. 12: **Hybrid$_\mathbf{3}$** of pseudorandomness proof.

```
F.Setup($1^\lambda$)
$(\mathsf{ck}_1^*, \mathsf{td}) \leftarrow \mathsf{COM}.\mathcal{E}_1(1^\lambda)$
$\mathsf{ck}_2 \leftarrow \mathsf{COM.Setup}(1^\lambda)$
$\mathsf{crs}_1 \leftarrow \mathsf{NIZK}_1.\mathsf{Setup}(1^\lambda)$
$\mathsf{crs}_2^*, \tau_2 \leftarrow \mathsf{NIZK}_2.\mathcal{S}_1(1^\lambda)$
$\mathsf{pp} := (\mathsf{ck}_1^*, \mathsf{ck}_2, \mathsf{crs}_1, \mathsf{crs}_2^*)$
return pp
```

```
F.KeyGen(pp)
parse $\mathsf{pp} := (\mathsf{ck}_1^*, \mathsf{ck}_2, \mathsf{crs}_1, \mathsf{crs}_2^*)$
$\mathbf{k} \xleftarrow{\$} \chi_k^m$
$\mathbf{c}_k^* \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_2, \mathbf{0}_m; \rho_k)$
return $(\mathsf{pk} := \mathbf{c}_k^*, \mathsf{sk} := \mathbf{k})$
```

```
F.Eval(sk, $t, x$)
parse $\mathsf{sk} := \mathbf{k}$
$\mathbf{B}_x := \mathcal{G}(t, x) \in R_q^{h \times m}$
$\mathbf{z} := \lfloor \mathbf{B}_x \mathbf{k} \rceil_p \in R_p^h$
$y := \mathsf{RO}_z(t, x, \mathbf{z})$
return $y$
```

```
F.Request(pp, pk, $t, x$)
parse $\mathsf{pp} := (\mathsf{ck}_1^*, \mathsf{ck}_2, \mathsf{crs}_1, \mathsf{crs}_2^*)$
$\mathbf{R} \xleftarrow{\$} \chi_r^{h \times (\ell+m)}$
$\mathbf{c}_r \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_1^*, (\mathbf{R}, x); \rho_r)$
$\mathbf{A}_r := \mathsf{RO}_r(t, \mathbf{c}_r) \in R_q^{(\ell+m) \times m}$
$\mathbf{B}_x := \mathcal{G}(t, x) \in R_q^{h \times m}$
$\mathbf{C}_x := \mathbf{R}\mathbf{A}_r + \mathbf{B}_x \in R_q^{h \times m}$
$\mathsf{stmt}_1 := (\mathbf{c}_r, \mathbf{C}_x, \mathsf{ck}_1, \mathbf{A}_r, \mathcal{G}, t)$
$\mathsf{wit}_1 := (\mathbf{R}, x, \rho_r)$
$\pi_c \leftarrow \mathsf{NIZK}_1.\mathsf{P}(\mathsf{crs}_1, \mathsf{stmt}_1, \mathsf{wit}_1)$
$\mathsf{st} := (t, x, \mathbf{R}, \mathbf{C}_x, \mathsf{pk}, \mathbf{A}_r)$
$\mathsf{req} := (\mathbf{c}_r, \mathbf{C}_x, \pi_c)$
return (st, req)
```

```
F.BlindEval(pp, pk, sk, $t$, req)
parse $\mathsf{pp} := (\mathsf{ck}_1^*, \mathsf{ck}_2, \mathsf{crs}_1, \mathsf{crs}_2^*)$
parse $\mathsf{pk} := \mathbf{c}_k^*, \mathsf{sk} := \mathbf{k}$
parse $\mathsf{req} := (\mathbf{c}_r, \mathbf{C}_x, \pi_c)$
$\mathbf{A}_r := \mathsf{RO}_r(t, \mathbf{c}_r) \in R_q^{(\ell+m) \times m}$
if $\mathsf{NIZK}_1.\mathsf{V}(\mathsf{crs}_1, \pi_c, \mathsf{stmt}_1) \neq 1$ then
    return $\perp$
$(\mathbf{u}_x, \mathbf{v}_k) := \mathcal{T}[t, \mathbf{c}_r]$
$\mathsf{stmt}_2 := (\mathbf{u}_x, \mathbf{C}_x, \mathbf{c}_k^*, \mathbf{v}_k, \mathsf{ck}_2, \mathbf{A}_r)$
$\pi_s^* \leftarrow \mathsf{NIZK}_2.\mathcal{S}_2(\mathsf{crs}_2^*, \tau_2, \mathsf{stmt}_2)$
return $\mathsf{rep} := (\mathbf{u}_x, \mathbf{v}_k, \pi_s^*)$
```

```
F.Finalize(pp, rep, st)
parse $\mathsf{pp} := (\mathsf{ck}_1^*, \mathsf{ck}_2, \mathsf{crs}_1, \mathsf{crs}_2^*)$
parse $\mathsf{rep} := (\mathbf{u}_x, \mathbf{v}_k, \pi_s^*)$
parse $\mathsf{st} := (t, x, \mathbf{R}, \mathbf{C}_x, \mathsf{pk} := \mathbf{c}_k^*, \mathbf{A}_r)$
if $\mathsf{NIZK}_2.\mathsf{V}(\mathsf{crs}_2^*, \pi_s^*, \mathsf{stmt}_2) \neq 1$ then
    return $\perp$
$\mathbf{z} := \lfloor \mathbf{u}_x - \mathbf{R}\mathbf{v}_k \rceil_p \in R_p^h$    // $= \lfloor \mathbf{B}_x \mathbf{k} \rceil_p$
$y := \mathsf{RO}_z(t, x, \mathbf{z})$
return $y$
```

```
RO_z($t, x, \mathbf{z}$)
if $\mathcal{F}[t, x, \mathbf{z}] = \perp$ then
    $\mathcal{F}[t, x, \mathbf{z}] \xleftarrow{\$} \{0,1\}^\lambda$
return $\mathcal{F}[t, x, \mathbf{z}]$
```

```
RO_r($t, \mathbf{c}_r$)
if $\mathcal{H}[t, \mathbf{c}_r] = \perp$ then
    $\mathbf{A}_r := \mathcal{H}[t, \mathbf{c}_r] \xleftarrow{\$} R_q^{(\ell+m) \times m}$
    $(\mathbf{R}, x) \leftarrow \mathsf{COM}.\mathcal{E}_2(\mathsf{ck}_1^*, \mathsf{td}, \mathbf{c}_r)$
    $\mathbf{B}_x := \mathcal{G}(t, x) \in R_q^{h \times m}$
    $\mathbf{e}_s \xleftarrow{\$} \chi^{\ell+m}, \mathbf{e}'_s \xleftarrow{\$} \chi_1^h$
    $\mathbf{v}_k := \mathbf{A}_r \mathbf{k} + \mathbf{e}_s \in R_q^{\ell+m}$
    $\mathbf{C}_x := \mathbf{R}\mathbf{A}_r + \mathbf{B}_x \in R_q^{h \times m}$
    $\mathbf{u}_x := \mathbf{C}_x \mathbf{k} + \mathbf{e}'_s \in R_q^h$
    $\mathcal{T}[t, \mathbf{c}_r] := (\mathbf{u}_x, \mathbf{v}_k)$
return $\mathcal{H}[t, \mathbf{c}_r]$
```

Fig. 13: **Hybrid$_\mathbf{4}$** of pseudorandomness proof.

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

```
$(\mathsf{ck}_1^*, \mathsf{td}) \leftarrow \mathsf{COM}.\mathcal{E}_1(1^\lambda)$
$\mathsf{ck}_2 \leftarrow \mathsf{COM.Setup}(1^\lambda)$
$\mathsf{crs}_1 \leftarrow \mathsf{NIZK}_1.\mathsf{Setup}(1^\lambda)$
$\mathsf{crs}_2^*, \tau_2 \leftarrow \mathsf{NIZK}_2.\mathcal{S}_1(1^\lambda)$
$\mathsf{pp} := (\mathsf{ck}_1^*, \mathsf{ck}_2, \mathsf{crs}_1, \mathsf{crs}_2^*)$
$\textbf{return } \mathsf{pp}$
```

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

```
$\textbf{parse } \mathsf{pp} := (\mathsf{ck}_1^*, \mathsf{ck}_2, \mathsf{crs}_1, \mathsf{crs}_2^*)$
$\mathbf{c}_k^* \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_2, \mathbf{0}_m; \rho_k)$
$\textbf{return } (\mathsf{pk} := \mathbf{c}_k^*, \mathsf{sk} := \bot)$
```

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

```
$\textbf{if } \mathcal{Z}[t, x] = \bot \textbf{ then}$
    $\mathbf{z}' \xleftarrow{\$} R_q^h$
    $\mathcal{Z}[t, x] := \mathbf{z}'$
$\mathbf{z} := \lfloor \mathcal{Z}[t, x] \rceil_p$
$y := \mathsf{RO}_z(t, x, \mathbf{z})$
$\textbf{return } y$
```

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

```
$\textbf{parse } \mathsf{pp} := (\mathsf{ck}_1^*, \mathsf{ck}_2, \mathsf{crs}_1, \mathsf{crs}_2^*)$
$\mathbf{R} \xleftarrow{\$} \chi_r^{h \times (\ell + m)}$
$\mathbf{c}_r \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_1^*, (\mathbf{R}, x); \rho_r)$
$\mathbf{A}_r := \mathsf{RO}_r(t, \mathbf{c}_r) \in R_q^{(\ell + m) \times m}$
$\mathbf{B}_x := \mathcal{G}(t, x) \in R_q^{h \times m}$
$\mathbf{C}_x := \mathbf{R}\mathbf{A}_r + \mathbf{B}_x \in R_q^{h \times m}$
$\mathsf{stmt}_1 := (\mathbf{c}_r, \mathbf{C}_x, \mathsf{ck}_1, \mathbf{A}_r, \mathcal{G}, t)$
$\mathsf{wit}_1 := (\mathbf{R}, x, \rho_r)$
$\pi_c \leftarrow \mathsf{NIZK}_1.\mathsf{P}(\mathsf{crs}_1, \mathsf{stmt}_1, \mathsf{wit}_1)$
$\mathsf{st} := (t, x, \mathbf{R}, \mathbf{C}_x, \mathsf{pk}, \mathbf{A}_r)$
$\mathsf{req} := (\mathbf{c}_r, \mathbf{C}_x, \pi_c)$
$\textbf{return } (\mathsf{st}, \mathsf{req})$
```

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

```
$\textbf{parse } \mathsf{pp} := (\mathsf{ck}_1^*, \mathsf{ck}_2, \mathsf{crs}_1, \mathsf{crs}_2^*)$
$\textbf{parse } \mathsf{pk} := \mathbf{c}_k^*, \mathsf{sk} := \bot$
$\textbf{parse } \mathsf{req} := (\mathbf{c}_r, \mathbf{C}_x, \pi_c)$
$\mathbf{A}_r := \mathsf{RO}_r(t, \mathbf{c}_r) \in R_q^{(\ell + m) \times m}$
$\textbf{if } \mathsf{NIZK}_1.\mathsf{V}(\mathsf{crs}_1, \pi_c, \mathsf{stmt}_1) \neq 1 \textbf{ then}$
    $\textbf{return } \bot$
$(\mathbf{u}_x, \mathbf{v}_k) := \mathcal{T}[t, \mathbf{c}_r]$
$\mathsf{stmt}_2 := (\mathbf{u}_x, \mathbf{C}_x, \mathbf{c}_k^*, \mathbf{v}_k, \mathsf{ck}_2, \mathbf{A}_r)$
$\pi_s^* \leftarrow \mathsf{NIZK}_2.\mathcal{S}_2(\mathsf{crs}_2^*, \tau_2, \mathsf{stmt}_2)$
$\textbf{return } \mathsf{rep} := (\mathbf{u}_x, \mathbf{v}_k, \pi_s^*)$
```

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

```
$\textbf{parse } \mathsf{pp} := (\mathsf{ck}_1^*, \mathsf{ck}_2, \mathsf{crs}_1, \mathsf{crs}_2^*)$
$\textbf{parse } \mathsf{rep} := (\mathbf{u}_x, \mathbf{v}_k, \pi_s^*)$
$\textbf{parse } \mathsf{st} := (t, x, \mathbf{R}, \mathbf{C}_x, \mathsf{pk} := \mathbf{c}_k^*, \mathbf{A}_r)$
$\textbf{if } \mathsf{NIZK}_2.\mathsf{V}(\mathsf{crs}_2^*, \pi_s^*, \mathsf{stmt}_2) \neq 1 \textbf{ then}$
    $\textbf{return } \bot$
$\mathbf{z} := \lfloor \mathbf{u}_x - \mathbf{R}\mathbf{v}_k \rceil_p \in R_p^h$   $/\!\!/ = \lfloor \mathbf{B}_x \mathbf{k} \rceil_p$
$y := \mathsf{RO}_z(t, x, \mathbf{z})$
$\textbf{return } y$
```

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

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

$\mathsf{RO}_r(t, \mathbf{c}_r)$

```
$\textbf{if } \mathcal{H}[t, \mathbf{c}_r] = \bot \textbf{ then}$
    $\mathbf{A}_r := \mathcal{H}[t, \mathbf{c}_r] \xleftarrow{\$} R_q^{(\ell + m) \times m}$
    $(\mathbf{R}, x) \leftarrow \mathsf{COM}.\mathcal{E}_2(\mathsf{ck}_1^*, \mathsf{td}, \mathbf{c}_r)$
    $\mathbf{v}_k \xleftarrow{\$} R_q^{\ell + m}$
    $\textbf{if } \mathcal{Z}[t, x] = \bot \textbf{ then}$
        $\mathbf{z}' \xleftarrow{\$} R_q^h$
        $\mathcal{Z}[t, x] := \mathbf{z}'$
    $\widetilde{\mathbf{e}}' \xleftarrow{\$} \chi_1^h, \widetilde{\mathbf{e}} \xleftarrow{\$} \chi^{\ell + m}$
    $\mathbf{e}^* := \widetilde{\mathbf{e}}' - \mathbf{R}\widetilde{\mathbf{e}} \in R^h$
    $\mathbf{u}_x := \mathbf{R}\mathbf{v}_k + \mathcal{Z}[t, x] + \mathbf{e}^* \in R_q^h$
    $\mathcal{T}[t, \mathbf{c}_r] := (\mathbf{u}_x, \mathbf{v}_k)$
$\textbf{return } \mathcal{H}[t, \mathbf{c}_r]$
```

Fig. 14: $\mathbf{Hybrid_5}$ of pseudorandomness proof.

In order to show that $\mathbf{Hybrid_4} \approx \mathbf{Hybrid_5}$ we construct an adversary $\mathcal{B}_5$ against the $\mathsf{iMLWE\text{-}RU\text{-}R}_{\mathsf{param}}$ assumption, for $\mathsf{param} := (\mathcal{G}, Q_\mathsf{M}, Q_\mathsf{P}, Q_x^\infty, q, m, \ell + m, h, 2\lambda, \beta_r, p, \bar{\chi})$. For answering Eval queries (t, x), $\mathcal{B}_5$ makes an oracle query $\mathbf{z} \leftarrow \mathsf{Samp\text{-}PRF}(t||x)$ to $\mathsf{iMLWE\text{-}RU\text{-}R}$ challenger, and returns $y := \mathsf{RO}_z(t, x, \mathbf{z})$. For answering RO_r queries $(t, \mathbf{c}_r)$, $\mathcal{B}_5$ uses the commitment trapdoor to extract $(\mathbf{R}, x) := \mathsf{COM}.\mathcal{E}_2(\mathsf{ck}_1^*, \mathsf{td}, \mathbf{c}_r)$ (as in the previous hybrid), and makes an oracle call $(\mathbf{A}_r, \mathbf{v}_k, \mathbf{u}_x) \leftarrow \mathsf{SampMLWE\text{-}RU}(\mathbf{R}, t||x)$ to the $\mathsf{iMLWE\text{-}RU\text{-}R}$ challenger. Then, $\mathcal{B}_5$ sets $\mathcal{H}[t, \mathbf{c}_r] := \mathbf{A}_r$ and stores $\mathcal{T}[t, \mathbf{c}_r] := (\mathbf{u}_x, \mathbf{v}_k)$. The rest of the queries are answered as in the previous hybrid.

If the $\mathsf{iMLWE\text{-}RU\text{-}R}$ challenger used $b = 1$, then the view of $\mathcal{A}$ simulated by $\mathcal{B}_5$ is exactly as in $\mathbf{Hybrid_4}$, and if it used $b = 0$, then the view of $\mathcal{A}$ is as in $\mathbf{Hybrid_5}$. Therefore, if $\mathcal{A}$ can distinguish between the two hybrids with non-negligible advantage, then $\mathcal{B}_5$ can break the $\mathsf{iMLWE\text{-}RU\text{-}R}_{\mathsf{param}}$ assumption, where Q_M and Q_P in the $\mathsf{iMLWE\text{-}RU\text{-}R}$ assumption denote the number of Eval and RO_r queries, and Q_x^∞ denotes the maximum number of identical $(t||x)$ queries, respectively, that the adversary $\mathcal{A}$ makes. Hence, it follows that

$$\left|\mathsf{Adv}_{\mathcal{A},4}(\lambda) - \mathsf{Adv}_{\mathcal{A},5}(\lambda)\right| \leq \mathsf{Adv}_{\mathcal{B}_5}^{\mathsf{iMLWE}_{\mathsf{param}}}(\lambda),$$

and in particular $\mathbf{Hybrid_4} \approx \mathbf{Hybrid_5}$.

$\mathbf{Hybrid_6}$: In this hybrid we replace the Eval algorithm with a random oracle, which is depicted in Figure 15. This corresponds to the pseudorandomness game $\mathsf{POPRF}^0_{\mathsf{F},\mathcal{A},\mathcal{S},\mathsf{RO}_r,\mathsf{RO}_z}$ (Definition 3).

Let BAD_2 be the event that the adversary $\mathcal{A}$ queries $(t, x, \mathbf{z})$ to RO_z oracle, for some $\mathbf{z} := \lfloor \mathcal{Z}[t, x] \rceil_p$, before ever making a query to the RO_r oracle. Clearly, we have that $\mathbf{Hybrid_5}$ and $\mathbf{Hybrid_6}$ are identical until the event BAD_2 happens. We observe that the only distinguishing advantage the adversary $\mathcal{A}$ has between $\mathbf{Hybrid_5}$ and $\mathbf{Hybrid_6}$ is if $\mathcal{A}$ queries RO_z oracle with some input $(t, x, \mathbf{z})$, where initially it holds that $\mathbf{z} \neq \lfloor \mathcal{Z}[t, x] \rceil_p$, but after making a call to RO_r oracle, we have that it is set to $\mathcal{Z}[t, x] := \mathbf{z}'$, where $\mathbf{z} := \lfloor \mathbf{z}' \rceil_p$. In this case, $\mathcal{A}$ will obtain two different outputs for the same $(t, x, \mathbf{z})$ input to RO_z oracle (before and after making a call to RO_r oracle), and it can trivially distinguish the hybrids. The probability of this happening corresponds to the aforementioned BAD_2 event happening, which we now argue that it can only happen with a negligible probability.

Note that inside the RO_r oracle, we sample $\mathbf{v}_k$ and $\mathbf{z}'$ uniformly from $R_q^{\ell+m}$ and R_q^h, respectively. Then, we compute $\mathbf{u}_x := \mathbf{R}\mathbf{v}_k + \mathbf{z}' + \mathbf{e}^* \in R_q^h$, for some $\mathbf{e}^*$ random in R^h, and we set $\mathcal{Z}[t, x] := \mathbf{z}'$. Therefore, the only value that is under the control of the adversary is $\mathbf{R}$, 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 $\mathbf{R}\mathbf{v}_k$ will actually cancel out during the computation of $\lfloor \mathbf{u}_x - \mathbf{R}\mathbf{v}_k \rceil_p$ (in the $\mathsf{Finalize}$ algorithm). Since $\mathbf{z}'$ is sampled independently of $\mathbf{e}^*$ and after $\mathbf{R}$ is fixed by the adversary, the distribution of each coefficient of $\mathbf{z}' + \mathbf{e}^*$ 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 PPT algorithm, it can only make polynomially many queries to RO_z oracle. Let Q_z denote the total number of queries that $\mathcal{A}$ makes to RO_z oracle, then

$$\Pr[\mathsf{BAD}_2] \leq Q_z \cdot \left(\frac{1}{p} + \frac{1}{q}\right)^{dh} =: \epsilon_{\mathsf{BAD}}.$$

Hence, it follows that

$$\left|\mathsf{Adv}_{\mathcal{A},5}(\lambda) - \mathsf{Adv}_{\mathcal{A},6}(\lambda)\right| \leq \epsilon_{\mathsf{BAD}},$$

and in particular $\mathbf{Hybrid_5} \approx \mathbf{Hybrid_6}$.

Finally, we note that the simulator $\mathcal{S}$ answers the oracle calls as in $\mathbf{Hybrid_6}$, and putting everything together, we obtain

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

as claimed. This completes the proof of Theorem 3. □

```
F.Setup(1^λ)
$(\mathsf{ck}_1^*, \mathsf{td}) \leftarrow \mathsf{COM}.\mathcal{E}_1(1^\lambda)$
$\mathsf{ck}_2 \leftarrow \mathsf{COM.Setup}(1^\lambda)$
$\mathsf{crs}_1 \leftarrow \mathsf{NIZK}_1.\mathsf{Setup}(1^\lambda)$
$\mathsf{crs}_2^*, \tau_2 \leftarrow \mathsf{NIZK}_2.\mathcal{S}_1(1^\lambda)$
$\mathsf{pp} \coloneqq (\mathsf{ck}_1^*, \mathsf{ck}_2, \mathsf{crs}_1, \mathsf{crs}_2^*)$
return $\mathsf{pp}$
```

```
F.KeyGen(pp)
parse $\mathsf{pp} \coloneqq (\mathsf{ck}_1^*, \mathsf{ck}_2, \mathsf{crs}_1, \mathsf{crs}_2^*)$
$\mathbf{c}_k^* \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_2, \mathbf{0}_m; \rho_k)$
return $(\mathsf{pk} \coloneqq \mathbf{c}_k^*, \mathsf{sk} \coloneqq \bot)$
```

```
F.Eval(sk, t, x)
if $\mathcal{Y}[t, x] = \bot$ then
    $\mathcal{Y}[t, x] \xleftarrow{\$} \{0,1\}^\lambda$
return $\mathcal{Y}[t, x]$
```

```
F.Request(pp, pk, t, x)
parse $\mathsf{pp} \coloneqq (\mathsf{ck}_1^*, \mathsf{ck}_2, \mathsf{crs}_1, \mathsf{crs}_2^*)$
$\mathbf{R} \xleftarrow{\$} \chi_r^{h \times (\ell+m)}$
$\mathbf{c}_r \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_1^*, (\mathbf{R}, x); \rho_r)$
$\mathbf{A}_r \coloneqq \mathsf{RO}_r(t, \mathbf{c}_r) \in R_q^{(\ell+m) \times m}$
$\mathbf{B}_x \coloneqq \mathcal{G}(t, x) \in R_q^{h \times m}$
$\mathbf{C}_x \coloneqq \mathbf{R}\mathbf{A}_r + \mathbf{B}_x \in R_q^{h \times m}$
$\mathsf{stmt}_1 \coloneqq (\mathbf{c}_r, \mathbf{C}_x, \mathsf{ck}_1, \mathbf{A}_r, \mathcal{G}, t)$
$\mathsf{wit}_1 \coloneqq (\mathbf{R}, x, \rho_r)$
$\pi_c \leftarrow \mathsf{NIZK}_1.\mathsf{P}(\mathsf{crs}_1, \mathsf{stmt}_1, \mathsf{wit}_1)$
$\mathsf{st} \coloneqq (t, x, \mathbf{R}, \mathbf{C}_x, \mathsf{pk}, \mathbf{A}_r)$
$\mathsf{req} \coloneqq (\mathbf{c}_r, \mathbf{C}_x, \pi_c)$
return $(\mathsf{st}, \mathsf{req})$
```

```
F.BlindEval(pp, pk, sk, t, req)
parse $\mathsf{pp} \coloneqq (\mathsf{ck}_1^*, \mathsf{ck}_2, \mathsf{crs}_1, \mathsf{crs}_2^*)$
parse $\mathsf{pk} \coloneqq \mathbf{c}_k^*, \mathsf{sk} \coloneqq \bot$
parse $\mathsf{req} \coloneqq (\mathbf{c}_r, \mathbf{C}_x, \pi_c)$
$\mathbf{A}_r \coloneqq \mathsf{RO}_r(t, \mathbf{c}_r) \in R_q^{(\ell+m) \times m}$
if $\mathsf{NIZK}_1.\mathsf{V}(\mathsf{crs}_1, \pi_c, \mathsf{stmt}_1) \neq 1$ then
    return $\bot$
$(\mathbf{u}_x, \mathbf{v}_k) \coloneqq \mathcal{T}[t, \mathbf{c}_r]$
$\mathsf{stmt}_2 \coloneqq (\mathbf{u}_x, \mathbf{C}_x, \mathbf{c}_k^*, \mathbf{v}_k, \mathsf{ck}_2, \mathbf{A}_r)$
$\pi_s^* \leftarrow \mathsf{NIZK}_2.\mathcal{S}_2(\mathsf{crs}_2^*, \tau_2, \mathsf{stmt}_2)$
return $\mathsf{rep} \coloneqq (\mathbf{u}_x, \mathbf{v}_k, \pi_s^*)$
```

```
F.Finalize(pp, rep, st)
parse $\mathsf{pp} \coloneqq (\mathsf{ck}_1^*, \mathsf{ck}_2, \mathsf{crs}_1, \mathsf{crs}_2^*)$
parse $\mathsf{rep} \coloneqq (\mathbf{u}_x, \mathbf{v}_k, \pi_s^*)$
parse $\mathsf{st} \coloneqq (t, x, \mathbf{R}, \mathbf{C}_x, \mathsf{pk} \coloneqq \mathbf{c}_k^*, \mathbf{A}_r)$
if $\mathsf{NIZK}_2.\mathsf{V}(\mathsf{crs}_2^*, \pi_s^*, \mathsf{stmt}_2) \neq 1$ then
    return $\bot$
$\mathbf{z} \coloneqq \lfloor \mathbf{u}_x - \mathbf{R}\mathbf{v}_k \rceil_p \in R_p^h$    // $= \lfloor \mathbf{B}_x \mathbf{k} \rceil_p$
$y \coloneqq \mathsf{RO}_z(t, x, \mathbf{z})$
return $y$
```

```
RO_z(t, x, z)
if $\mathcal{F}[t, x, \mathbf{z}] = \bot$ then
    if $\mathcal{Z}[t, x] \neq \bot$ and $\mathbf{z} = \lfloor \mathcal{Z}[t, x] \rceil_p$ then
        if $\mathcal{Y}[t, x] = \bot$ then
            $\mathcal{Y}[t, x] \xleftarrow{\$} \{0,1\}^\lambda$
        $\mathcal{F}[t, x, \mathbf{z}] \coloneqq \mathcal{Y}[t, x]$
    else
        $\mathcal{F}[t, x, \mathbf{z}] \xleftarrow{\$} \{0,1\}^\lambda$
return $\mathcal{F}[t, x, \mathbf{z}]$
```

```
RO_r(t, c_r)
if $\mathcal{H}[t, \mathbf{c}_r] = \bot$ then
    $\mathbf{A}_r \coloneqq \mathcal{H}[t, \mathbf{c}_r] \xleftarrow{\$} R_q^{(\ell+m) \times m}$
    $(\mathbf{R}, x) \leftarrow \mathsf{COM}.\mathcal{E}_2(\mathsf{ck}_1^*, \mathsf{td}, \mathbf{c}_r)$
    $\mathbf{v}_k \xleftarrow{\$} R_q^{\ell+m}$
    if $\mathcal{Z}[t, x] = \bot$ then
        $\mathbf{z}' \xleftarrow{\$} R_q^h$
        $\mathcal{Z}[t, x] \coloneqq \mathbf{z}'$
    $\widetilde{\mathbf{e}}' \xleftarrow{\$} \chi_1^h$, $\widetilde{\mathbf{e}} \xleftarrow{\$} \chi^{\ell+m}$
    $\mathbf{e}^* \coloneqq \widetilde{\mathbf{e}}' - \mathbf{R}\widetilde{\mathbf{e}} \in R^h$
    $\mathbf{u}_x \coloneqq \mathbf{R}\mathbf{v}_k + \mathcal{Z}[t, x] + \mathbf{e}^* \in R_q^h$
    $\mathcal{T}[t, \mathbf{c}_r] \coloneqq (\mathbf{u}_x, \mathbf{v}_k)$
return $\mathcal{H}[t, \mathbf{c}_r]$
```

Fig. 15: **Hybrid$_6$** of pseudorandomness proof.

B.4 Request Privacy Analysis

Theorem 4. *The POPRF construction* F *from Figure 8 satisfies the request privacy against malicious servers (POPRIV2), given in Definition 4, with random oracles* RO_r *and* RO_z*, if:*

- *The client argument system* NIZK_1 *is computationally zero-knowledge and the server argument system* NIZK_2 *is computationally sound (Definition 17),*
- *The commitment scheme* COM *is computationally hiding and extractable (Definitions 14 and 16), 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* $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$*,* $\mathcal{B}_4$ *and* $\mathcal{B}_5$ *against the computational zero-knowledge of* NIZK_1*, computational hiding of* COM*, computational soundness of* NIZK_2*, extractability of* COM *and* $\mathsf{knMLWE}_{m,\ell+m,h,\chi}$ *assumption, respectively, such that*

$$\mathsf{Adv}^{\text{po-priv-2}}_{\mathsf{F},\mathcal{A},\mathsf{RO}_r,\mathsf{RO}_z}(\lambda) \leq \mathsf{Adv}^{\mathsf{CZK}}_{\mathsf{NIZK}_1,\mathcal{B}_1}(\lambda) + 2Q_{\mathsf{R}} \cdot \mathsf{Adv}^{\mathsf{CH}}_{\mathsf{COM},\mathcal{B}_2}(\lambda) + 2Q_{\mathsf{F}} \cdot \mathsf{Adv}^{\mathsf{CS}}_{\mathsf{NIZK}_2,\mathcal{B}_3}(\lambda)$$
$$+ Q^{\mathsf{pk}}_{\mathsf{R}} \cdot \mathsf{Adv}^{\mathsf{Ext}}_{\mathsf{COM},\mathcal{B}_4}(\lambda) + 2Q_{\mathsf{R}} \cdot \mathsf{Adv}^{\mathsf{knMLWE}_{\ell+m,m,h,\chi}}_{\mathcal{B}_5}(\lambda),$$

where Q_{R} *and* Q_{F} *denote the number of* $\mathsf{Request}$ *and* $\mathsf{Finalize}$ *oracle queries, respectively, and* $Q^{\mathsf{pk}}_{\mathsf{R}}$ *denotes the number of queries to* $\mathsf{Request}$ *with different* 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 oracles $\mathsf{Request}, \mathsf{Finalize}$ and the random oracles RO_r and 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 also denote $|\Pr[\mathbf{Hybrid_i} = 1] - \Pr[\mathbf{Hybrid_{i+1}} = 1]| \leq \mathsf{negl}(\lambda)$ by $\mathbf{Hybrid_i} \approx \mathbf{Hybrid_{i+1}}$. We consider the hybrids listed below, and for each hybrid we depict the changes using figures. The figures include the whole protocol for ease of reference to remind how the real execution works, but we emphasize that we do not modify the adversarially controlled algorithms and parts.

$\mathbf{Hybrid_0}$: 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 Figure 8. Hence, it follows that

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

$\mathbf{Hybrid_1}$: In this hybrid, we replace crs_1 and proof π_c with a simulated CRS and proof $(\mathsf{crs}_1^*, \pi_c^*)$, as shown in Figure 16.

In order to show that $\mathbf{Hybrid_0} \approx \mathbf{Hybrid_1}$, we construct a reduction $\mathcal{B}_1$ to the computational zero-knowledge property of 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 $\mathcal{B}_1$ prior to passing the public parameters pp to $\mathcal{A}$, it receives a CRS crs_1^* from its zero-knowledge challenger, and sets $\mathsf{crs}_1 := \mathsf{crs}_1^*$, instead of generating crs_1 via the $\mathsf{NIZK}_1.\mathsf{Setup}$ algorithm. When $\mathcal{A}$ makes a $\mathsf{Request}$ query $(\mathsf{pk}, t, x_0, x_1)$, $\mathcal{B}_1$ proceeds as in $\mathbf{Hybrid_0}$ to compute $\mathbf{c}_{r,i}$ and $\mathbf{C}_{x_i}$, for $i \in \{0,1\}$, but it makes two oracle calls to its zero-knowledge challenger with the inputs $((\mathbf{c}_{r,i}, \mathbf{C}_{x_i}, \mathbf{A}_{r,1,i}, \mathcal{G}, t), (\mathbf{R}_i, x_i))$, for $i \in \{0,1\}$, to obtain the proofs $(\pi^*_{c,0}, \pi^*_{c,1})$. Finally, $\mathcal{B}_1$ responds to the $\mathsf{Request}$ query with $(\mathsf{req}_b := (\mathbf{c}_{r,b}, \mathbf{C}_{x_b}, \pi^*_{c,b}), \mathsf{req}_{1-b} := (\mathbf{c}_{r,1-b}, \mathbf{C}_{x_{1-b}}, \pi^*_{c,1-b}))$.

If the zero-knowledge challenger of NIZK_1 used the honest setup and prover algorithms to generate 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 NIZK_1. Hence, it follows that

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

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

```
$\mathsf{F.KeyGen}(\mathsf{pp})$
$\textbf{parse } \mathsf{pp} \coloneqq (\mathsf{ck}_1, \mathsf{ck}_2, \mathsf{crs}_1^*, \mathsf{crs}_2)$
$\mathbf{k} \xleftarrow{\$} \chi_k^m$
$\mathbf{c}_k \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_2, \mathbf{k}; \rho_k)$
$\textbf{return } (\mathsf{pk} \coloneqq \mathbf{c}_k, \mathsf{sk} \coloneqq \mathbf{k})$
```

```
$\mathsf{F.Request}(\mathsf{pp}, \mathsf{pk}, t, x)$
$\textbf{parse } \mathsf{pp} \coloneqq (\mathsf{ck}_1, \mathsf{ck}_2, \mathsf{crs}_1^*, \mathsf{crs}_2)$
$\mathbf{R} \xleftarrow{\$} \chi_r^{h \times (\ell+m)}$
$\mathbf{c}_r \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_1, (\mathbf{R}, x); \rho_r)$
$\mathbf{A}_r \coloneqq \mathsf{RO}_r(t, \mathbf{c}_r) \in R_q^{(\ell+m) \times m}$
$\mathbf{B}_x \coloneqq \mathcal{G}(t, x) \in R_q^{h \times m}$
$\mathbf{C}_x \coloneqq \mathbf{R}\mathbf{A}_r + \mathbf{B}_x \in R_q^{h \times m}$
$\mathsf{stmt}_1 \coloneqq (\mathbf{c}_r, \mathbf{C}_x, \mathsf{ck}_1, \mathbf{A}_r, \mathcal{G}, t)$
$\pi_c^* \leftarrow \mathsf{NIZK}_1.\mathcal{S}_2(\mathsf{crs}_1^*, \tau, \mathsf{stmt}_1)$
$\mathsf{st} \coloneqq (t, x, \mathbf{R}, \mathbf{C}_x, \mathsf{pk}, \mathbf{A}_r)$
$\mathsf{req} \coloneqq (\mathbf{c}_r, \mathbf{C}_x, \pi_c^*)$
$\textbf{return } (\mathsf{st}, \mathsf{req})$
```

```
$\mathsf{F.BlindEval}(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}, t, \mathsf{req})$
$\textbf{parse } \mathsf{pp} \coloneqq (\mathsf{ck}_1, \mathsf{ck}_2, \mathsf{crs}_1^*, \mathsf{crs}_2)$
$\textbf{parse } \mathsf{pk} \coloneqq \mathbf{c}_k, \mathsf{sk} \coloneqq \mathbf{k}$
$\textbf{parse } \mathsf{req} \coloneqq (\mathbf{c}_r, \mathbf{C}_x, \pi_c^*)$
$\mathbf{A}_r \coloneqq \mathsf{RO}_r(t, \mathbf{c}_r) \in R_q^{(\ell+m) \times m}$
$\textbf{if } \mathsf{NIZK}_1.\mathsf{V}(\mathsf{crs}_1^*, \pi_c^*, \mathsf{stmt}_1) \neq 1 \textbf{ then}$
    $\textbf{return } \bot$
$\mathbf{e}_s \xleftarrow{\$} \chi^{\ell+m}, \mathbf{e}'_s \xleftarrow{\$} \chi_1^h$
$\mathbf{v}_k \coloneqq \mathbf{A}_r\mathbf{k} + \mathbf{e}_s \in R_q^{\ell+m}$
$\mathbf{u}_x \coloneqq \mathbf{C}_x\mathbf{k} + \mathbf{e}'_s \in R_q^h$
$\mathsf{stmt}_2 \coloneqq (\mathbf{u}_x, \mathbf{C}_x, \mathbf{c}_k, \mathbf{v}_k, \mathsf{ck}_2, \mathbf{A}_r)$
$\mathsf{wit}_2 \coloneqq (\mathbf{k}, \mathbf{e}, \mathbf{e}_s, \mathbf{e}'_s, \rho_k)$
$\pi_s \leftarrow \mathsf{NIZK}_2.\mathsf{P}(\mathsf{crs}_2, \mathsf{stmt}_2, \mathsf{wit}_2)$
$\textbf{return } \mathsf{rep} \coloneqq (\mathbf{u}_x, \mathbf{v}_k, \pi_s)$
```

```
$\mathsf{F.Finalize}(\mathsf{pp}, \mathsf{rep}, \mathsf{st})$
$\textbf{parse } \mathsf{pp} \coloneqq (\mathsf{ck}_1, \mathsf{ck}_2, \mathsf{crs}_1^*, \mathsf{crs}_2)$
$\textbf{parse } \mathsf{rep} \coloneqq (\mathbf{u}_x, \mathbf{v}_k, \pi_s)$
$\textbf{parse } \mathsf{st} \coloneqq (t, x, \mathbf{R}, \mathbf{C}_x, \mathsf{pk} \coloneqq \mathbf{c}_k, \mathbf{A}_r)$
$\textbf{if } \mathsf{NIZK}_2.\mathsf{V}(\mathsf{crs}_2, \pi_s, \mathsf{stmt}_2) \neq 1 \textbf{ then}$
    $\textbf{return } \bot$
$\mathbf{z} \coloneqq \lfloor \mathbf{u}_x - \mathbf{R}\mathbf{v}_k \rceil_p \in R_p^h$    $/\!/ = \lfloor \mathbf{B}_x\mathbf{k} \rceil_p$
$y \coloneqq \mathsf{RO}_z(t, x, \mathbf{z})$
$\textbf{return } y$
```

```
$\mathsf{F.Eval}(\mathsf{sk}, t, x)$
$\textbf{parse } \mathsf{sk} \coloneqq \mathbf{k}$
$\mathbf{B}_x \coloneqq \mathcal{G}(t, x) \in R_q^{h \times m}$
$\mathbf{z} \coloneqq \lfloor \mathbf{B}_x\mathbf{k} \rceil_p \in R_p^h$
$y \coloneqq \mathsf{RO}_z(t, x, \mathbf{z})$
$\textbf{return } y$
```

```
$\mathsf{RO}_r(t, \mathbf{c}_r)$
$\textbf{if } \mathcal{H}[t, \mathbf{c}_r] = \bot \textbf{ then}$
    $\mathcal{H}[t, \mathbf{c}_r] \xleftarrow{\$} R_q^{(\ell+m) \times m}$
$\textbf{return } \mathcal{H}[t, \mathbf{c}_r]$
```

```
$\mathsf{RO}_z(t, x, \mathbf{z})$
$\textbf{if } \mathcal{F}[t, x, \mathbf{z}] = \bot \textbf{ then}$
    $\mathcal{F}[t, x, \mathbf{z}] \xleftarrow{\$} \{0,1\}^\lambda$
$\textbf{return } \mathcal{F}[t, x, \mathbf{z}]$
```

Fig. 16: $\mathbf{Hybrid_1}$ of request privacy proof.

and in particular $\mathbf{Hybrid_0} \approx \mathbf{Hybrid_1}$.

$\mathbf{Hybrid_2}$: In this hybrid, we commit to $\mathbf{c}_r^* \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_1, \mathbf{0}_{h\cdot(\ell+m)+1})$, where $\mathbf{0}_{h\cdot(\ell+m)+1}$ is an all zero vector of length $h \cdot (\ell + m) + 1$, instead of computing the commitment $\mathbf{c}_r \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_1, (\mathbf{R}, x))$ as in the previous hybrid. This change is shown in Figure 17.

In order to show that $\mathbf{Hybrid_1} \approx \mathbf{Hybrid_2}$, 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 $\mathbf{Hybrid_1}$, with the exception of the $\mathsf{Request}$ queries. Let Q_{R} denote the number of $\mathsf{Request}$ queries that the adversary $\mathcal{A}$ makes. Upon receiving a $\mathsf{Request}$ query $(\mathsf{pk}, t, x_0, x_1)$, $\mathcal{B}_2$ samples $\mathbf{R}_i \overset{\$}{\leftarrow} \chi_r^{h\times(\ell+m)}$, for $i \in \{0, 1\}$, sets $(m_{i,0} \coloneqq (\mathbf{R}_i, x_i), m_{i,1} \coloneqq \mathbf{0}_{h\cdot(\ell+m)+1})$, and submits the pairs $(m_{i,0}, m_{i,1})$ 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 to compute $\mathbf{C}_{x,i}$ as in $\mathbf{Hybrid_1}$ and to simulates $\pi_{c,i}$, for $i \in \{0, 1\}$. Finally, $\mathcal{B}_2$ responds to the $\mathsf{Request}$ query with $(\mathsf{req}_b \coloneqq (\mathbf{c}_{r,b}, \mathbf{C}_{x_b}, \pi_{c,b}), \mathsf{req}_{1-b} \coloneqq (\mathbf{c}_{r,1-b}, \mathbf{C}_{x_{1-b}}, \pi_{c,1-b}))$.

If the computational hiding challenger of COM responds with a commitment to $m_{i,0}$, then we are exactly in $\mathbf{Hybrid_1}$, and if it responds with a commitment to $m_{i,1}$, then we are in $\mathbf{Hybrid_2}$. 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 2-challenge variant) and the adversary $\mathcal{A}$ makes at most Q_{R} queries to the $\mathsf{Request}$ oracle, it follows by a standard argument that

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

and in particular $\mathbf{Hybrid_1} \approx \mathbf{Hybrid_2}$.

$\mathbf{Hybrid_3}$: Let BAD be the event that for a $\mathsf{Finalize}$ query $(j, \mathsf{rep}_0 \coloneqq (\mathbf{u}_{x,0}, \mathbf{v}_{k,0}, \pi_{s,0}), \mathsf{rep}_1 \coloneqq (\mathbf{u}_{x,1}, \mathbf{v}_{k,1}, \pi_{s,1}))$, it holds that either

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

where $\mathbf{C}_{x,j}$ and $\mathbf{A}_{r,j}$ are values obtained from the j-th query to $\mathsf{Request}$ oracle. We will prove that the BAD event can only happen with negligible probability due to the soundness of NIZK_2 arguments system. Looking ahead, we need here such a reduction to the soundness of NIZK_2 because later in $\mathbf{Hybrid_4}$ 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 $\mathsf{pk} \coloneqq \mathbf{c}_k$ is well-formed.

Clearly, we have that $\mathbf{Hybrid_2}$ and $\mathbf{Hybrid_3}$ are identical until the event BAD happens. Hence, we show that we can bound the probability $\Pr[\mathsf{BAD}]$ by constructing a reduction $\mathcal{B}_3$ to the computational soundness of NIZK_2 argument system. We note that all queries are answered as in $\mathbf{Hybrid_2}$. Let Q_{F} denote the number of $\mathsf{Finalize}$ queries that $\mathcal{A}$ performs. $\mathcal{B}_3$ randomly chooses a $q^* \in [2Q_{\mathsf{F}}]$ to serve as its guess for the the proof π_s that will be forged by the adversary $\mathcal{A}$ (note that each $\mathsf{Finalize}$ query includes two proofs, hence the guess from $[2Q_{\mathsf{F}}]$). Let $(j, \mathsf{rep}_0 \coloneqq (\mathbf{u}_{x,0}, \mathbf{v}_{k,0}, \pi_{s,0}), \mathsf{rep}_1 \coloneqq (\mathbf{u}_{x,1}, \mathbf{v}_{k,1}, \pi_{s,1}))$ be the j-th query to $\mathsf{Finalize}$, where $j = \lceil q^*/2 \rceil$, 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 $\mathbf{k}, \mathbf{e}_s, \mathbf{e}'_s$ or ρ_k (which are computationally infeasible to compute), and instead just guesses which of the proofs would have caused the event BAD to happen. Hence, we have that

$$\left|\mathsf{Adv}_{\mathcal{A},2}(\lambda) - \mathsf{Adv}_{\mathcal{A},3}(\lambda)\right| \leq 2Q_{\mathsf{F}} \cdot \mathsf{Adv}^{\mathsf{CS}}_{\mathsf{NIZK}_2,\mathcal{B}_3}(\lambda),$$

and in particular $\mathbf{Hybrid_2} \approx \mathbf{Hybrid_3}$.

$\mathbf{Hybrid_4}$: In this hybrid, we replace the commitment key 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 $\mathsf{sk} \coloneqq \mathbf{k}$ from the public key $\mathsf{pk} \coloneqq \mathbf{c}_k$, i.e., run $\mathbf{k} \leftarrow \mathsf{COM}.\mathcal{E}_2(\mathsf{ck}_2^*, \mathsf{td}, \mathbf{c}_k)$, and store $\mathbf{k}$ for later use. These changes are depicted in Figure 18.

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

```
$\mathsf{F.KeyGen}(\mathsf{pp})$
$\textbf{parse } \mathsf{pp} \coloneqq (\mathsf{ck}_1, \mathsf{ck}_2, \mathsf{crs}_1^*, \mathsf{crs}_2)$
$\mathbf{k} \xleftarrow{\$} \chi_k^m$
$\mathbf{c}_k \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_2, \mathbf{k}; \rho_k)$
$\textbf{return } (\mathsf{pk} \coloneqq \mathbf{c}_k, \mathsf{sk} \coloneqq \mathbf{k})$
```

```
$\mathsf{F.Request}(\mathsf{pp}, \mathsf{pk}, t, x)$
$\textbf{parse } \mathsf{pp} \coloneqq (\mathsf{ck}_1, \mathsf{ck}_2, \mathsf{crs}_1^*, \mathsf{crs}_2)$
$\mathbf{R} \xleftarrow{\$} \chi_r^{h \times (\ell+m)}$
$\mathbf{c}_r^* \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_1, \mathbf{0}_{h\cdot(\ell+m)+1}; \rho_r)$
$\mathbf{A}_r \coloneqq \mathsf{RO}_r(t, \mathbf{c}_r^*) \in R_q^{(\ell+m)\times m}$
$\mathbf{B}_x \coloneqq \mathcal{G}(t, x) \in R_q^{h \times m}$
$\mathbf{C}_x \coloneqq \mathbf{R}\mathbf{A}_r + \mathbf{B}_x \in R_q^{h \times m}$
$\mathsf{stmt}_1 \coloneqq (\mathbf{c}_r^*, \mathbf{C}_x, \mathsf{ck}_1, \mathbf{A}_r, \mathcal{G}, t)$
$\pi_c^* \leftarrow \mathsf{NIZK}_1.\mathcal{S}_2(\mathsf{crs}_1^*, \tau, \mathsf{stmt}_1)$
$\mathsf{st} \coloneqq (t, x, \mathbf{R}, \mathbf{C}_x, \mathsf{pk}, \mathbf{A}_r)$
$\mathsf{req} \coloneqq (\mathbf{c}_r^*, \mathbf{C}_x, \pi_c^*)$
$\textbf{return } (\mathsf{st}, \mathsf{req})$
```

```
$\mathsf{F.BlindEval}(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}, t, \mathsf{req})$
$\textbf{parse } \mathsf{pp} \coloneqq (\mathsf{ck}_1, \mathsf{ck}_2, \mathsf{crs}_1^*, \mathsf{crs}_2)$
$\textbf{parse } \mathsf{pk} \coloneqq \mathbf{c}_k, \mathsf{sk} \coloneqq \mathbf{k}$
$\textbf{parse } \mathsf{req} \coloneqq (\mathbf{c}_r^*, \mathbf{C}_x, \pi_c^*)$
$\mathbf{A}_r \coloneqq \mathsf{RO}_r(t, \mathbf{c}_r^*) \in R_q^{(\ell+m)\times m}$
$\textbf{if } \mathsf{NIZK}_1.\mathsf{V}(\mathsf{crs}_1^*, \pi_c^*, \mathsf{stmt}_1) \neq 1 \textbf{ then}$
    $\textbf{return } \bot$
$\mathbf{e}_s \xleftarrow{\$} \chi^{\ell+m}, \mathbf{e}_s' \xleftarrow{\$} \chi_1^h$
$\mathbf{v}_k \coloneqq \mathbf{A}_r\mathbf{k} + \mathbf{e}_s \in R_q^{\ell+m}$
$\mathbf{u}_x \coloneqq \mathbf{C}_x\mathbf{k} + \mathbf{e}_s' \in R_q^h$
$\mathsf{stmt}_2 \coloneqq (\mathbf{u}_x, \mathbf{C}_x, \mathbf{c}_k, \mathbf{v}_k, \mathsf{ck}_2, \mathbf{A}_r)$
$\mathsf{wit}_2 \coloneqq (\mathbf{k}, \mathbf{e}, \mathbf{e}_s, \mathbf{e}_s', \rho_k)$
$\pi_s \leftarrow \mathsf{NIZK}_2.\mathsf{P}(\mathsf{crs}_2, \mathsf{stmt}_2, \mathsf{wit}_2)$
$\textbf{return } \mathsf{rep} \coloneqq (\mathbf{u}_x, \mathbf{v}_k, \pi_s)$
```

```
$\mathsf{F.Finalize}(\mathsf{pp}, \mathsf{rep}, \mathsf{st})$
$\textbf{parse } \mathsf{pp} \coloneqq (\mathsf{ck}_1, \mathsf{ck}_2, \mathsf{crs}_1^*, \mathsf{crs}_2)$
$\textbf{parse } \mathsf{rep} \coloneqq (\mathbf{u}_x, \mathbf{v}_k, \pi_s)$
$\textbf{parse } \mathsf{st} \coloneqq (t, x, \mathbf{R}, \mathbf{C}_x, \mathsf{pk} \coloneqq \mathbf{c}_k, \mathbf{A}_r)$
$\textbf{if } \mathsf{NIZK}_2.\mathsf{V}(\mathsf{crs}_2, \pi_s, \mathsf{stmt}_2) \neq 1 \textbf{ then}$
    $\textbf{return } \bot$
$\mathbf{z} \coloneqq \lfloor \mathbf{u}_x - \mathbf{R}\mathbf{v}_k \rceil_p \in R_p^h$    $/\!\!/ = \lfloor \mathbf{B}_x\mathbf{k} \rceil_p$
$y \coloneqq \mathsf{RO}_z(t, x, \mathbf{z})$
$\textbf{return } y$
```

```
$\mathsf{F.Eval}(\mathsf{sk}, t, x)$
$\textbf{parse } \mathsf{sk} \coloneqq \mathbf{k}$
$\mathbf{B}_x \coloneqq \mathcal{G}(t, x) \in R_q^{h \times m}$
$\mathbf{z} \coloneqq \lfloor \mathbf{B}_x\mathbf{k} \rceil_p \in R_p^h$
$y \coloneqq \mathsf{RO}_z(t, x, \mathbf{z})$
$\textbf{return } y$
```

```
$\mathsf{RO}_r(t, \mathbf{c}_r)$
$\textbf{if } \mathcal{H}[t, \mathbf{c}_r] = \bot \textbf{ then}$
    $\mathcal{H}[t, \mathbf{c}_r] \xleftarrow{\$} R_q^{(\ell+m)\times m}$
$\textbf{return } \mathcal{H}[t, \mathbf{c}_r]$
```

```
$\mathsf{RO}_z(t, x, \mathbf{z})$
$\textbf{if } \mathcal{F}[t, x, \mathbf{z}] = \bot \textbf{ then}$
    $\mathcal{F}[t, x, \mathbf{z}] \xleftarrow{\$} \{0, 1\}^\lambda$
$\textbf{return } \mathcal{F}[t, x, \mathbf{z}]$
```

Fig. 17: $\mathbf{Hybrid_2}$ of request privacy proof.

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

```
$\mathsf{ck}_1 \leftarrow \mathsf{COM.Setup}(1^\lambda)$
$(\mathsf{ck}_2^*, \mathsf{td}) \leftarrow \mathsf{COM}.\mathcal{E}_1(1^\lambda)$
$(\mathsf{crs}_1^*, \tau) \leftarrow \mathsf{NIZK}_1.\mathcal{S}_1(1^\lambda)$
$\mathsf{crs}_2 \leftarrow \mathsf{NIZK}_2.\mathsf{Setup}(1^\lambda)$
$\mathsf{pp} \coloneqq (\mathsf{ck}_1, \mathsf{ck}_2^*, \mathsf{crs}_1^*, \mathsf{crs}_2)$
$\mathbf{return}\ \mathsf{pp}$
```

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

```
$\mathbf{parse}\ \mathsf{pp} \coloneqq (\mathsf{ck}_1, \mathsf{ck}_2^*, \mathsf{crs}_1^*, \mathsf{crs}_2)$
$\mathbf{k} \xleftarrow{\$} \chi_k^m$
$\mathbf{c}_k \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_2^*, \mathbf{k}; \rho_k)$
$\mathbf{return}\ (\mathsf{pk} \coloneqq \mathbf{c}_k, \mathsf{sk} \coloneqq \mathbf{k})$
```

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

```
$\mathbf{parse}\ \mathsf{pp} \coloneqq (\mathsf{ck}_1, \mathsf{ck}_2^*, \mathsf{crs}_1^*, \mathsf{crs}_2)$
$\mathbf{R} \xleftarrow{\$} \chi_r^{h \times (\ell+m)}$
$\mathbf{c}_r^* \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_1, \mathbf{0}_{h\cdot(\ell+m)+1}; \rho_r)$
$\mathbf{A}_r \coloneqq \mathsf{RO}_r(t, \mathbf{c}_r^*) \in R_q^{(\ell+m)\times m}$
$\mathbf{B}_x \coloneqq \mathcal{G}(t, x) \in R_q^{h \times m}$
$\mathbf{C}_x \coloneqq \mathbf{R}\mathbf{A}_r + \mathbf{B}_x \in R_q^{h \times m}$
$\mathsf{stmt}_1 \coloneqq (\mathbf{c}_r^*, \mathbf{C}_x, \mathsf{ck}_1, \mathbf{A}_r, \mathcal{G}, t)$
$\pi_c^* \leftarrow \mathsf{NIZK}_1.\mathcal{S}_2(\mathsf{crs}_1^*, \tau, \mathsf{stmt}_1)$
$\mathsf{st} \coloneqq (t, x, \mathbf{R}, \mathbf{C}_x, \mathsf{pk}, \mathbf{A}_r)$
$\mathsf{req} \coloneqq (\mathbf{c}_r^*, \mathbf{C}_x, \pi_c^*)$
$\mathbf{return}\ (\mathsf{st}, \mathsf{req})$
```

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

```
$\mathbf{parse}\ \mathsf{pp} \coloneqq (\mathsf{ck}_1, \mathsf{ck}_2^*, \mathsf{crs}_1^*, \mathsf{crs}_2)$
$\mathbf{parse}\ \mathsf{pk} \coloneqq \mathbf{c}_k, \mathsf{sk} \coloneqq \mathbf{k}$
$\mathbf{parse}\ \mathsf{req} \coloneqq (\mathbf{c}_r^*, \mathbf{C}_x, \pi_c^*)$
$\mathbf{A}_r \coloneqq \mathsf{RO}_r(t, \mathbf{c}_r^*) \in R_q^{(\ell+m)\times m}$
$\mathbf{if}\ \mathsf{NIZK}_1.\mathsf{V}(\mathsf{crs}_1^*, \pi_c^*, \mathsf{stmt}_1) \neq 1\ \mathbf{then}$
    $\mathbf{return}\ \perp$
$\mathbf{e}_s \xleftarrow{\$} \chi^{\ell+m}, \mathbf{e}_s' \xleftarrow{\$} \chi_1^h$
$\mathbf{v}_k \coloneqq \mathbf{A}_r\mathbf{k} + \mathbf{e}_s \in R_q^{\ell+m}$
$\mathbf{u}_x \coloneqq \mathbf{C}_x\mathbf{k} + \mathbf{e}_s' \in R_q^h$
$\mathsf{stmt}_2 \coloneqq (\mathbf{u}_x, \mathbf{C}_x, \mathbf{c}_k, \mathbf{v}_k, \mathsf{ck}_2^*, \mathbf{A}_r)$
$\mathsf{wit}_2 \coloneqq (\mathbf{k}, \mathbf{e}, \mathbf{e}_s, \mathbf{e}_s'.\rho_k)$
$\pi_s \leftarrow \mathsf{NIZK}_2.\mathsf{P}(\mathsf{crs}_2, \mathsf{stmt}_2, \mathsf{wit}_2)$
$\mathbf{return}\ \mathsf{rep} \coloneqq (\mathbf{u}_x, \mathbf{v}_k, \pi_s)$
```

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

```
$\mathbf{parse}\ \mathsf{pp} \coloneqq (\mathsf{ck}_1, \mathsf{ck}_2^*, \mathsf{crs}_1^*, \mathsf{crs}_2)$
$\mathbf{parse}\ \mathsf{rep} \coloneqq (\mathbf{u}_x, \mathbf{v}_k, \pi_s)$
$\mathbf{parse}\ \mathsf{st} \coloneqq (t, x, \mathbf{R}, \mathbf{C}_x, \mathsf{pk} \coloneqq \mathbf{c}_k, \mathbf{A}_r)$
$\mathbf{if}\ \mathsf{NIZK}_2.\mathsf{V}(\mathsf{crs}_2, \pi_s, \mathsf{stmt}_2) \neq 1\ \mathbf{then}$
    $\mathbf{return}\ \perp$
$\mathbf{if}\ \mathcal{K}[\mathsf{pk}] = \perp\ \mathbf{then}$
    $\mathbf{k} \leftarrow \mathsf{COM}.\mathcal{E}_2(\mathsf{ck}_2^*, \mathsf{td}, \mathbf{c}_k)$
    $\mathcal{K}[\mathsf{pk}] \coloneqq \mathbf{k}$
$\mathbf{z} \coloneqq \lfloor \mathbf{u}_x - \mathbf{R}\mathbf{v}_k \rceil_p \in R_p^h \quad /\!\!/ = \lfloor \mathbf{B}_x\mathbf{k} \rceil_p$
$y \coloneqq \mathsf{RO}_z(t, x, \mathbf{z})$
$\mathbf{return}\ y$
```

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

```
$\mathbf{parse}\ \mathsf{sk} \coloneqq \mathbf{k}$
$\mathbf{B}_x \coloneqq \mathcal{G}(t, x) \in R_q^{h \times m}$
$\mathbf{z} \coloneqq \lfloor \mathbf{B}_x\mathbf{k} \rceil_p \in R_p^h$
$y \coloneqq \mathsf{RO}_z(t, x, \mathbf{z})$
$\mathbf{return}\ y$
```

$\mathsf{RO}_r(t, \mathbf{c}_r)$

```
$\mathbf{if}\ \mathcal{H}[t, \mathbf{c}_r] = \perp\ \mathbf{then}$
    $\mathcal{H}[t, \mathbf{c}_r] \xleftarrow{\$} R_q^{(\ell+m)\times m}$
$\mathbf{return}\ \mathcal{H}[t, \mathbf{c}_r]$
```

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

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

Fig. 18: $\mathbf{Hybrid_4}$ of request privacy proof.

In order to prove that $\mathbf{Hybrid_3} \approx \mathbf{Hybrid_4}$ we construct an adversary $\mathcal{B}_4$ to the extractability property of COM. We note that all queries are answered as in $\mathbf{Hybrid_3}$, 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 ck_2 via the $\mathsf{COM.Setup}$ algorithm. Let Q_R^pk denote the number of $\mathsf{Request}$ queries that the adversary $\mathcal{A}$ makes with different pk inputs. For each such query $(\mathsf{pk}, t, x_0, x_1)$, upon $\mathcal{B}_4$ receiving a $\mathsf{Finalize}$ query $(j, \mathsf{rep}_0, \mathsf{rep}_1)$, where j corresponds to one of the Q_R^pk queries made to the $\mathsf{Request}$ oracle, $\mathcal{B}_4$ computes $\mathsf{sk} := \mathbf{k} \leftarrow \mathsf{COM}.\mathcal{E}_2(\mathsf{ck}_2^*, \mathsf{td}, \mathsf{pk} := \mathbf{c}_k)$, and stores it in the table $\mathcal{K}[\mathsf{pk}] := \mathbf{k}$ (for later use). The rest of the interaction is identical to $\mathbf{Hybrid_3}$.

Due to the soundness of the server proof π_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 $\mathcal{B}_4$ can break the extractability property of COM. Moreover, since the adversary $\mathcal{A}$ makes at most Q_R^pk queries to $\mathsf{Request}$ oracle, it follows that

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

and in particular $\mathbf{Hybrid_3} \approx \mathbf{Hybrid_4}$.

$\mathbf{Hybrid_5}$: In this hybrid, we replace $\mathbf{C}_x := \mathbf{R}\mathbf{A}_r + \mathbf{B}_x \in R_q^{h \times m}$ with $\mathbf{C}_x^* := \mathbf{U} + \mathbf{B}_x \in R_q^{h \times m}$ for a uniformly random matrix $\mathbf{U} \overset{\$}{\leftarrow} R_q^{h \times m}$, as shown in Figure 19.

In order to prove that $\mathbf{Hybrid_4} \approx \mathbf{Hybrid_5}$ we construct and adversary $\mathcal{B}_5$ against the $\mathsf{knMLWE}_{\ell+m,m,h,\chi}$ assumption. We note that the public parameters pp and responses to all queries are computed as in $\mathbf{Hybrid_4}$ with the exception of the $\mathsf{Request}$ queries. For each $\mathsf{Request}$ query $(\mathsf{pk}, t, x_0, x_1)$, $\mathcal{B}_5$ receives from the $\mathsf{knMLWE}_{\ell+m,m,h,\chi}$ challenger the pairs $(\mathbf{A}_i, \mathbf{U}_i)$, which it uses to compute $\mathbf{C}_{x,i}^* := \mathbf{U}_i + \mathcal{G}(t, x_i)$, for $i \in \{0,1\}$. Note that $\mathbf{C}_{x,i}^*$ forms part of the request that $\mathcal{B}_5$ returns to $\mathcal{A}$ as output of the $\mathsf{Request}$ query. Moreover, $\mathcal{B}_5$ programs the random oracle RO_r, such that on input the pair $(t, \mathbf{c}_{r,i}^*)$ 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 18, 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 $\mathsf{Finalize}$ query $(j, \mathsf{rep}_0 := (\mathbf{u}_{x,0}, \mathbf{v}_{k,0}, \pi_{s,0}), \mathsf{rep}_1 := (\mathbf{u}_{x,0}, \mathbf{v}_{k,0}, \pi_{s,0}))$ from $\mathcal{A}$, $\mathcal{B}_5$ proceeds as in the previous hybrid, to obtain $\mathsf{pk} := \mathbf{c}_k$ from the state st_j, and then extract $\mathsf{sk} := \mathbf{k}$ from pk using the commitment trapdoor. Additionally, $\mathcal{B}_5$ uses the secret key $\mathbf{k}$ to compute the values $\mathbf{z}_i^* := \lfloor \mathbf{u}_{x,i} - \mathbf{U}_i\mathbf{k} - \mathbf{R}_i(\mathbf{v}_{k,i} - \mathbf{A}_{r,i}\mathbf{k}) \rceil_p$ for $i \in \{0,1\}$, which constitutes an alternative way of computing the correct $\mathbf{z}_i^*$ value with the help of the secret key $\mathbf{k}$ and the randomness $\mathbf{R}_i$ that was sampled during the $\mathsf{Request}$ query. Finally, $\mathcal{B}$ computes the output y_i 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., $\mathbf{U}_i := \mathbf{R}_i\mathbf{A}_i$, for some $\mathbf{R}_i \in R_q^{h\times(\ell+m)}$, then the view is as in $\mathbf{Hybrid_4}$, and if it provided a uniformly random matrix $\mathbf{U}_i \in R_q^{h\times m}$, then the view is as in $\mathbf{Hybrid_5}$. Therefore, if $\mathcal{A}$ can distinguish between the two hybrids with non-negligible advantage, then $\mathcal{B}_5$ 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_R queries to the $\mathsf{Request}$ oracle, it follows by a standard argument that

$$\left|\mathsf{Adv}_{\mathcal{A},4}(\lambda) - \mathsf{Adv}_{\mathcal{A},5}(\lambda)\right| \leq 2Q_\mathsf{R} \cdot \mathsf{Adv}_{\mathcal{B}_5}^{\mathsf{knMLWE}_{\ell+m,m,h,\chi}}(\lambda),$$

and in particular $\mathbf{Hybrid_4} \approx \mathbf{Hybrid_5}$.

We note that at this point the transcript observed by the adversary $\mathcal{A}$ is independent of the challenge bit b, and hence $\mathsf{Adv}_{\mathcal{A},5}(\lambda) = 1/2$. Putting everything together, we obtain

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

as claimed. This completes the proof of Theorem 4. □

```
$\mathsf{F.Setup}(1^\lambda)$
$\mathsf{ck}_1 \leftarrow \mathsf{COM.Setup}(1^\lambda)$
$(\mathsf{ck}_2^*, \mathsf{td}) \leftarrow \mathsf{COM}.\mathcal{E}_1(1^\lambda)$
$(\mathsf{crs}_1^*, \tau) \leftarrow \mathsf{NIZK}_1.\mathcal{S}_1(1^\lambda)$
$\mathsf{crs}_2 \leftarrow \mathsf{NIZK}_2.\mathsf{Setup}(1^\lambda)$
$\mathsf{pp} \coloneqq (\mathsf{ck}_1, \mathsf{ck}_2^*, \mathsf{crs}_1^*, \mathsf{crs}_2)$
$\mathbf{return}\ \mathsf{pp}$
```

```
$\mathsf{F.KeyGen}(\mathsf{pp})$
$\mathbf{parse}\ \mathsf{pp} \coloneqq (\mathsf{ck}_1, \mathsf{ck}_2^*, \mathsf{crs}_1^*, \mathsf{crs}_2)$
$\mathbf{k} \xleftarrow{\$} \chi_k^m$
$\mathbf{c}_k \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_2^*, \mathbf{k}; \rho_k)$
$\mathbf{return}\ (\mathsf{pk} \coloneqq \mathbf{c}_k, \mathsf{sk} \coloneqq \mathbf{k})$
```

```
$\mathsf{F.Request}(\mathsf{pp}, \mathsf{pk}, t, x)$
$\mathbf{parse}\ \mathsf{pp} \coloneqq (\mathsf{ck}_1, \mathsf{ck}_2^*, \mathsf{crs}_1^*, \mathsf{crs}_2)$
$\mathbf{R} \xleftarrow{\$} \chi_r^{h \times (\ell+m)}$
$\mathbf{c}_r^* \leftarrow \mathsf{COM.Commit}(\mathsf{ck}_1, \mathbf{0}_{h\cdot(\ell+m)+1}; \rho_r)$
$\mathbf{A}_r \coloneqq \mathsf{RO}_r(t, \mathbf{c}_r^*) \in R_q^{(\ell+m)\times m}$
$\mathbf{B}_x \coloneqq \mathcal{G}(t, x) \in R_q^{h\times m}$
$\mathbf{U} \xleftarrow{\$} R_q^{h\times m}$
$\mathbf{C}_x^* \coloneqq \mathbf{U} + \mathbf{B}_x \in R_q^{h\times m}$
$\mathsf{stmt}_1 \coloneqq (\mathbf{c}_r^*, \mathbf{C}_x,^* \mathsf{ck}_1, \mathbf{A}_r, \mathcal{G}, t)$
$\pi_c^* \leftarrow \mathsf{NIZK}_1.\mathcal{S}_2(\mathsf{crs}_1^*, \tau, \mathsf{stmt}_1)$
$\mathsf{st} \coloneqq (t, x, \mathbf{R}, \mathbf{C}_x^*, \mathsf{pk}, \mathbf{A}_r)$
$\mathsf{req} \coloneqq (\mathbf{c}_r^*, \mathbf{C}_x^*, \pi_c^*)$
$\mathbf{return}\ (\mathsf{st}, \mathsf{req})$
```

```
$\mathsf{F.BlindEval}(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}, t, \mathsf{req})$
$\mathbf{parse}\ \mathsf{pp} \coloneqq (\mathsf{ck}_1, \mathsf{ck}_2^*, \mathsf{crs}_1^*, \mathsf{crs}_2)$
$\mathbf{parse}\ \mathsf{pk} \coloneqq \mathbf{c}_k, \mathsf{sk} \coloneqq \mathbf{k}$
$\mathbf{parse}\ \mathsf{req} \coloneqq (\mathbf{c}_r^*, \mathbf{C}_x^*, \pi_c^*)$
$\mathbf{A}_r \coloneqq \mathsf{RO}_r(t, \mathbf{c}_r^*) \in R_q^{(\ell+m)\times m}$
$\mathbf{if}\ \mathsf{NIZK}_1.\mathsf{V}(\mathsf{crs}_1^*, \pi_c^*, \mathsf{stmt}_1) \neq 1\ \mathbf{then}$
    $\mathbf{return}\ \perp$
$\mathbf{e}_s \xleftarrow{\$} \chi^{\ell+m}, \mathbf{e}_s' \xleftarrow{\$} \chi_1^h$
$\mathbf{v}_k \coloneqq \mathbf{A}_r\mathbf{k} + \mathbf{e}_s \in R_q^{\ell+m}$
$\mathbf{u}_x \coloneqq \mathbf{C}_x^*\mathbf{k} + \mathbf{e}_s' \in R_q^h$
$\mathsf{stmt}_2 \coloneqq (\mathbf{u}_x, \mathbf{C}_x^*, \mathbf{c}_k, \mathbf{v}_k, \mathsf{ck}_2^*, \mathbf{A}_r)$
$\mathsf{wit}_2 \coloneqq (\mathbf{k}, \mathbf{e}, \mathbf{e}_s, \mathbf{e}_s', \rho_k)$
$\pi_s \leftarrow \mathsf{NIZK}_2.\mathsf{P}(\mathsf{crs}_2, \mathsf{stmt}_2, \mathsf{wit}_2)$
$\mathbf{return}\ \mathsf{rep} \coloneqq (\mathbf{u}_x, \mathbf{v}_k, \pi_s)$
```

```
$\mathsf{F.Finalize}(\mathsf{pp}, \mathsf{rep}, \mathsf{st})$
$\mathbf{parse}\ \mathsf{pp} \coloneqq (\mathsf{ck}_1, \mathsf{ck}_2^*, \mathsf{crs}_1^*, \mathsf{crs}_2)$
$\mathbf{parse}\ \mathsf{rep} \coloneqq (\mathbf{u}_x, \mathbf{v}_k, \pi_s)$
$\mathbf{parse}\ \mathsf{st} \coloneqq (t, x, \mathbf{R}, \mathbf{C}_x^*, \mathsf{pk} \coloneqq \mathbf{c}_k, \mathbf{A}_r)$
$\mathbf{if}\ \mathsf{NIZK}_2.\mathsf{V}(\mathsf{crs}_2, \pi_s, \mathsf{stmt}_2) \neq 1\ \mathbf{then}$
    $\mathbf{return}\ \perp$
$\mathbf{if}\ \mathcal{K}[\mathsf{pk}] = \perp\ \mathbf{then}$
    $\mathbf{k} \leftarrow \mathsf{COM}.\mathcal{E}_2(\mathsf{ck}_2^*, \mathsf{td}, \mathbf{c}_k)$
    $\mathcal{K}[\mathsf{pk}] \coloneqq \mathbf{k}$
$\mathbf{z}^* \coloneqq \lfloor \mathbf{u}_x - \mathbf{U}\mathcal{K}[\mathsf{pk}] - \mathbf{R}(\mathbf{v}_k - \mathbf{A}_r\mathbf{k}) \rceil_p \in R_p^h$    $/\!/ = \lfloor \mathbf{B}_x\mathbf{k} \rceil_p$
$y \coloneqq \mathsf{RO}_z(t, x, \mathbf{z}^*)$
$\mathbf{return}\ y$
```

```
$\mathsf{F.Eval}(\mathsf{sk}, t, x)$
$\mathbf{parse}\ \mathsf{sk} \coloneqq \mathbf{k}$
$\mathbf{B}_x \coloneqq \mathcal{G}(t, x) \in R_q^{h\times m}$
$\mathbf{z} \coloneqq \lfloor \mathbf{B}_x\mathbf{k} \rceil_p \in R_p^h$
$y \coloneqq \mathsf{RO}_z(t, x, \mathbf{z})$
$\mathbf{return}\ y$
```

```
$\mathsf{RO}_r(t, \mathbf{c}_r)$
$\mathbf{if}\ \mathcal{H}[t, \mathbf{c}_r] = \perp\ \mathbf{then}$
    $\mathcal{H}[t, \mathbf{c}_r] \xleftarrow{\$} R_q^{(\ell+m)\times m}$
$\mathbf{return}\ \mathcal{H}[t, \mathbf{c}_r]$
```

```
$\mathsf{RO}_z(t, x, \mathbf{z})$
$\mathbf{if}\ \mathcal{F}[t, x, \mathbf{z}] = \perp\ \mathbf{then}$
    $\mathcal{F}[t, x, \mathbf{z}] \xleftarrow{\$} \{0,1\}^\lambda$
$\mathbf{return}\ \mathcal{F}[t, x, \mathbf{z}]$
```

Fig. 19: **Hybrid$_\mathbf{5}$** of request privacy proof.

C Issues in Prior Constructions and Models

C.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 π 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 sk. The authors attempt to solve this dilemma by requiring the server to give its 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 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 sk as part of the public key 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 B.4.

C.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)* 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 POUNIQ *is defined as follows:*

```
$\mathsf{POUNIQ}_{\mathsf{F},\mathcal{A},\mathsf{RO}}(\lambda)$
$R := [\cdot], q := 0$
$\mathsf{pp} \leftarrow \mathsf{F.Setup}(1^\lambda)$
$\mathsf{st}_{\mathsf{RO}} \leftarrow \mathsf{RO.Init}(1^\lambda)$
$(i, j, \mathsf{rep}_i, \mathsf{rep}_j) \leftarrow \mathcal{A}^{\mathsf{Request},\mathsf{RO}}(\mathsf{pp})$
assert $1 \leq i \leq j \leq q$
$(\mathsf{st}_i, \mathsf{pk}_i, t_i, x_i) := R[i]$
$(\mathsf{st}_j, \mathsf{pk}_j, t_j, x_j) := R[j]$
$y_i \leftarrow \mathsf{F.Finalize}^{\mathsf{RO}}_{\mathsf{pp}}(\mathsf{rep}_i, \mathsf{st}_i)$
$y_j \leftarrow \mathsf{F.Finalize}^{\mathsf{RO}}_{\mathsf{pp}}(\mathsf{rep}_j, \mathsf{st}_j)$
if $y_i = \perp \vee y_j = \perp$ then
    return 0
return $((\mathsf{pk}_i, t_i, x_i) = (\mathsf{pk}_j, t_j, x_j)) \wedge (y_i \neq y_j)$
```

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

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

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 $\mathsf{Wellformed}$ 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* $\mathsf{Wellformed}$ *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 $\mathsf{POKBIND}$ *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 8 satisfies key binding.

Theorem 5. *If* COM *is computationally binding, then the OPRF construction* F *from Figure 8 is key binding.*

Proof. First we define a $\mathsf{Wellformed}$ 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 $\mathsf{Wellformed}$ 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 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 8, 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 15) 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 6. *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*

$$\mathsf{Adv}^{\mathsf{po\text{-}uniq}}_{\mathsf{F},\mathcal{A},\mathsf{RO}}(\lambda) \leq \mathsf{Adv}^{\mathsf{po\text{-}key\text{-}bind}}_{\mathsf{F},\mathcal{B}_1,\mathsf{RO}}(\lambda) + \mathsf{Adv}^{\mathsf{po\text{-}priv\text{-}2}}_{\mathsf{F},\mathcal{B}_2,\mathsf{RO}}(\lambda)\,.$$

Proof. We note that the adversary $\mathcal{A}$ has access to the oracles $\mathsf{Request}$ 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

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

$\mathbf{Hybrid_1}$: Let BAD be the event that the adversary queries the $\mathsf{Request}$ 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 $\mathsf{Request}$ 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

$$\left|\mathsf{Adv}_{\mathcal{A},0}(\lambda) - \mathsf{Adv}_{\mathcal{A},1}(\lambda)\right| \leq \mathsf{Adv}^{\mathsf{po\text{-}key\text{-}bind}}_{\mathsf{F},\mathcal{B}_1,\mathsf{RO}}(\lambda)\,,$$

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 F. Analogous to the previous reduction, $\mathcal{B}_2$ receives the public parameters pp from the request privacy challenger, and forwards pp to $\mathcal{A}$. Moreover, RO queries of $\mathcal{A}$ are forwarded to the 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 (pk, t, x, x), received back a pair of request messages $(\mathsf{req}_0, \mathsf{req}_1)$ and sends 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 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 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 6. □

D More on the Message Mapping and NIZKs

D.1 Further Options for Message Mapping $\mathcal{G}$

Using the mapping from BLMR13 PRF [BLMR13] for $\mathcal{G}$. 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 (where the dimension parameter $h = m$ to match the secret key dimension in OPRF). 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

$$F^{\mathsf{blmr}}_{\mathbf{k}}(x) = \left\lfloor \prod_{i=0}^{L-1} \mathbf{A}_{x_i} \cdot \mathbf{k} \right\rceil_p. \tag{32}$$

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

$$\mathcal{G}^{\mathsf{blmr}} : x \mapsto \mathbf{B}_x = \prod_{i=0}^{L-1} \mathbf{A}_{x_i}. \tag{33}$$

For the client's NIZK proof π_c to be discussed in Section 5.3, 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.

Using a (symmetric) PRG for $\mathcal{G}$. Another option one may consider is to use a (symmetric) PRG to instantiate $\mathcal{G}$. 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 $\mathsf{Samp\text{-}PRF}$ oracle in $\mathsf{iMLWE\text{-}RU\text{-}R}$ would output samples (computationally) indistinguishable from MLWR samples (based on the PRG pseudorandomness property). One may then employ a suitable zk-SNARK system to instantiate the client's proof π_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 $\mathcal{G}$ as a random oracle. We also note the option of modeling $\mathcal{G}$ 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 $\mathsf{Samp\text{-}PRF}$ oracle in $\mathsf{iMLWE\text{-}RU\text{-}R}$ would simply output MLWR samples. Assuming $\mathcal{G}$ to be a random oracle would also reduce the control of an $\mathsf{iMLWE\text{-}RU\text{-}R}$ 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 preimage 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 $\mathcal{G}$ 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 $\mathcal{G}$ 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 efficiency, we believe that the main advantage of modelling $\mathcal{G}$ as a random oracle may arise in helping with the security reductions and reducing an adversary's winning chance in iMLWE-RU-R. We emphasize that modelling $\mathcal{G}$ as a random oracle is *not* be our main choice in this paper, and we merely present it as a possible option.

D.2 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 $\mathbf{B}_x$ is defined in (33). To be more specific, $\mathbf{B}_x = \prod_{i=0}^{L-1} \mathbf{A}_{x_i}$, 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 $\mathbf{B}_i := \prod_{j=0}^{i} \mathbf{A}_{x_i}$. 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 is present, we will have a similar situation as described above for the BP14 PRF, where some public matrix $\mathbf{B}_t$ will appear in the first expression describing $\mathbf{B}_0$.

E Instantiation of the Commitment Scheme

From our analyses in Section 4.2, 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.

For certain instantiations of the message mapping $\mathcal{G}$ (such as BMLR13 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 MLWE secret keys and $\mathbf{e}_{i,j}$'s are short error vectors where each entry is sampled from an error distribution χ_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 $\mathbf{c} = \mathbf{A}\mathbf{r}$. 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 $\mathbf{c} = \mathbf{A}\mathbf{r}$. 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[12] 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[13] 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[14] if $\mathsf{MLWE}_{w,n+M,\chi_e}$ is hard, and
2. (computationally) binding if $\mathsf{MSIS}_{n,n+M+w,2\beta_r}$ is hard where β_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 MSIS bound β_{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).

[12] Note that well-formedness of the commitment along with having small coefficients in error, randomness, and message will be proven via a NIZK proof.

[13] 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]).

[14] In fact, the commitment outputs are (computationally) indistinguishable from uniformly random vectors over R_q.

Extraction/decryption correctness. In our NIZK proofs employed in 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
$$\|q_m \mathbf{e}_{i,0}^\top \mathbf{r} + m_i\|_\infty < q/2.$$
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 MLWE_{n,χ_e} 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 χ_r as the entries of $\mathbf{R}$ in LeOPaRd as the client security already relies on $\mathsf{MLWE}_{\ell,\chi_r}$. 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 σ_0 as the server security already relies on 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).