Round-optimal Verifiable Oblivious Pseudorandom Functions from Ideal Lattices

Martin R. Albrecht[1], Alex Davidson[2], Amit Deo[1,3], and Nigel P. Smart[4,5]

[1] Information Security Group, Royal Holloway, University of London, UK
[2] Cloudflare Portugal
[3] ENS de Lyon, Laboratoire LIP (U. Lyon, CNRS, ENSL, INRIA, UCBL), France
[4] COSIC-imec, KU Leuven, Belgium
[5] Dept. Computer Science, University of Bristol, UK
martin.albrecht@royalholloway.ac.uk, alex.davidson92@gmail.com,
amit.deo@ens-lyon.fr, nigel.smart@kuleuven.be

Abstract. Verifiable Oblivious Pseudorandom Functions (VOPRFs) are protocols that allow a client to learn verifiable pseudorandom function (PRF) evaluations on inputs of their choice. The PRF evaluations are computed by a server using their own secret key. The security of the protocol prevents both the server from learning anything about the client's input, and likewise the client from learning anything about the server's key. VOPRFs have many applications including password-based authentication, secret-sharing, anonymous authentication and efficient private set intersection. In this work, we construct the first round-optimal (online) VOPRF protocol that retains security from well-known subexponential lattice hardness assumptions. Our protocol requires constructions of non-interactive zero-knowledge arguments of knowledge (NIZKAoK). Using recent developments in the area of post-quantum zero-knowledge arguments of knowledge, we show that our VOPRF may be securely instantiated in the quantum random oracle model. We construct such arguments as extensions of prior work in the area of lattice-based zero-knowledge proof systems.

1 Introduction

A verifiable oblivious pseudorandom function (VOPRF) is an interactive protocol between two parties; a client and a server. Intuitively, this protocol allows a server to provide a client with an evaluation of a pseudorandom function (PRF) on an input x chosen by the client using the server's key k. Informally, the security of a VOPRF, from the server's perspective, guarantees that the client learns nothing more than the PRF evaluated at x using k as the key where the server has committed to k in advance. Informally, security from the perspective of the client guarantees the three conditions below:

1. the server learns nothing about the input x;
2. the client's output in the protocol is indeed the evaluation on input x and key k;

The fact that the client is ensured that its output corresponds to the key committed to by the server makes the protocol a *verifiable* oblivious PRF. If we were to remove this requirement, the protocol would be an *oblivious* pseudorandom function (OPRF). From a multi-party computation perspective, an OPRF can be seen as a protocol that securely achieves the functionality $g(x, k) = (F_k(x), \perp)$ where F_k is a PRF using key k and $\perp$ indicates that the server receives no output. Applications of (V)OPRFs include secure keyword search [27], private set intersection [36], secure data de-duplication [38], password-protected secret sharing [33,34], password-authenticated key exchange (PAKE) [35] and privacy-preserving lightweight authentication mechanisms [21].

A number of these applications have had recent and considerable real-world impact. The work of Jarecki at al. [35] constructs a PAKE protocol, known as OPAQUE, using an OPRF as a core primitive. The OPAQUE protocol is intended for integration with TLS 1.3 to enable password-based authentication, and it is currently in the process of being standardised [39] by the Crypto Forum Research Group (CFRG)[6] as part of the PAKE selection process [17]. In addition, the work of Davidson et al. [21] constructs a privacy-preserving authorisation mechanism (known as Privacy Pass) for anonymously bypassing Internet reverse Turing tests based entirely on the security of a VOPRF. The Privacy Pass protocol is currently used at scale by the web performance company Cloudflare [57], and there have also been recent efforts to standardise the protocol design [22]. Both Privacy Pass and OPAQUE use discrete-log (DL) based (V)OPRF constructions to produce notably performant protocols. Finally, there is a separate and ongoing effort being carried forward by the CFRG [23] focusing directly on standardising performant DL-based VOPRF constructions.

Unfortunately, and in spite of the practical value of VOPRFs, all of the available constructions in the literature to date are based on classical assumptions, such as decisional Diffie-Hellman (DDH) and RSA. As such, all current VOPRFs would be insecure when confronted with an adversary that can run quantum computations. Therefore, the design of a post-quantum secure VOPRF is required to ensure that the applications above remain secure in these future adversarial conditions.[7] In fact, for full post-quantum security, both the PRF and the VOPRF protocol itself must be secure in the quantum adversarial model. While PRF constructions with claimed post-quantum security are standard, it remains an open problem to translate these into secure VOPRF protocols.

Constructions of PRFs arising from lattice-based cryptography originated from the work of Banerjee, Peikert and Rosen [6]. These constructions are post-quantum secure assuming the hardness of the learning with errors (LWE) problem against quantum adversaries [53]. To get around the fact that the LWE problem involves the addition of random small errors, carefully chosen rounding is used to obtain *deterministic* outputs for PRFs based on the LWE assumption [6,11,5]. These earlier works on LWE-based PRFs were followed by construc-

[6] A subsidiary of the Internet Research Task Force (IRTF).

[7] Note that using post-quantum secure VOPRF primitives in either the OPAQUE or Privacy Pass examples above would immediately result in PQ-secure alternatives.

tions of more advanced variants of PRFs [16,14,50]. Despite this, there is yet to be an OPRF protocol for any LWE-based PRF. The same is true for variants of these constructions based on the ring LWE (RLWE) problem [5].

Contributions. In this work, we instantiate a round-optimal[8] VOPRF whose security relies on subexponential hardness assumptions over lattices. Our construction assumes certain non-interactive zero-knowledge arguments of knowledge (NIZKAoKs). We use the protocol of Yang et al. [58] as an example instantiation of the required NIZKAoKs, to argue knowledge of inputs to the input-dependent part of PRF evaluations from the Banerjee and Peikert design [5] (henceforth BP14) in the ring setting. Alternatively, one can use Stern-like methods such as those in [41] and the recent protocol of Beullens [7]. These choices come with the advantage that results stating the validity of the Fiat-Shamir transform in the quantum random oracle model (QROM) [25,42] will apply.

We stress that our results show the *feasibility* of round-optimal VOPRF protocols based on lattice assumptions, rather than practicality. The performance of the VOPRF is negatively impacted by the required size of parameters (see Section 5.3). These parameters are necessary for instantiating our construction using reasonable underlying lattice assumptions – a consequence of using the BP14 PRF construction with our proof technique. Moreover, we require heavy zero-knowledge proof computations too ensure that neither participant deviates from the protocol. Some of these proofs may be removed by considering certain optimisations of our main protocol (see Section 3.2). Additionally, removing all zero-knowledge proofs and considering an honest-but-curious setting may result in a relatively efficient protocol.

Technical Overview. We design a VOPRF for a particular instantiation of the BP14 PRF in the ring setting. Specifically, for a particular *function* $\boldsymbol{a}^F : \{0,1\}^L \to R_q^{1\times\ell}$ where $R_q := \mathbb{Z}_q[X]/\langle X^n + 1\rangle$, we set out to design a VOPRF for the PRF

$$F_k(x) = \left\lfloor \frac{p}{q} \cdot \boldsymbol{a}^F(x) \cdot k \right\rceil$$

where the key $k \in R_q$ has small coefficients when represented in $\{-q/2, \ldots, q/2\}$, and $\lfloor \cdot \rceil$ represents rounding a rational to the nearest natural number. Our VOPRF protocol can be easily modified to handle other choices of $\boldsymbol{a}^F(x)$ (up to a change in parameter requirements). The security of this BP14 PRF construction can be reduced to the hardness of RLWE. Consider the PRF for 2-bit inputs: then $\boldsymbol{a}^F(x) = \boldsymbol{a}_1 \cdot G^{-1}(\boldsymbol{a}_2)$ where $\boldsymbol{a}_1, \boldsymbol{a}_2 \in R_q^{1\times\ell}$ are uniform and public, $G = (1, 2, \ldots, 2^{\ell-1})$ and $G^{-1}(\boldsymbol{a}_2) \in R_2^{\ell\times\ell}$ is binary. Very informally, for small

[8] Meaning that only two messages are sent in the online (query) phase.

$\boldsymbol{e}, \boldsymbol{e}'' \in R_q^{1\times\ell}$, uniform $\boldsymbol{e}' \in R_q^{1\times\ell}/(R_q \cdot G)$ and q much larger than p, we can write

$$\begin{aligned}
\left\lfloor \frac{p}{q} \cdot \boldsymbol{a}^F(x) \cdot k \right\rceil &= \left\lfloor \frac{p}{q} \cdot k \cdot \boldsymbol{a}_1 \cdot G^{-1}(\boldsymbol{a}_2) \right\rceil = \left\lfloor \frac{p}{q} \cdot (k \cdot \boldsymbol{a}_1 + \boldsymbol{e}) \cdot G^{-1}(\boldsymbol{a}_2) \right\rceil \\
&\approx_c \left\lfloor \frac{p}{q} \cdot (\boldsymbol{u}) \cdot G^{-1}(\boldsymbol{a}_2) \right\rceil \text{ (RLWE)} \\
&= \left\lfloor \frac{p}{q} (u'G + \boldsymbol{e}') \cdot G^{-1}(\boldsymbol{a}_2) \right\rceil = \left\lfloor \frac{p}{q} (u'\boldsymbol{a}_2 + \boldsymbol{e}'') + \frac{p}{q}\boldsymbol{e}' \cdot G^{-1}(\boldsymbol{a}_2) \right\rceil \\
&\approx_c \left\lfloor \frac{p}{q} \cdot \boldsymbol{u}'' + \frac{p}{q} \cdot \boldsymbol{e}' \cdot G^{-1}(\boldsymbol{a}_2) \right\rceil \text{ (RLWE)} \\
&= \left\lfloor \frac{p}{q} \cdot \widetilde{\boldsymbol{u}} \right\rceil
\end{aligned}$$

where $\boldsymbol{u}, \boldsymbol{u}'', \widetilde{\boldsymbol{u}}$ are uniform in $R_q^{1\times\ell}$ and u' is uniform in R_q. The proof of pseudorandomness builds on these ideas.

To provide intuition for our VOPRF design, we describe the rough form of our protocol (without zero-knowledge proofs). Given a public uniform $\boldsymbol{a} \in R_q^{1\times\ell}$, the high level overview is as follows:

1. The server publishes some commitment $\boldsymbol{c} := \boldsymbol{a} \cdot k + \boldsymbol{e}$ to a small key $k \in R_q$.
2. On input x, the client picks *small* $s \in R_q$, small $\boldsymbol{e}_1 \in R_q^{1\times\ell}$ and sends $\boldsymbol{c}_x = \boldsymbol{a} \cdot s + \boldsymbol{e}_1 + \boldsymbol{a}^F(x)$.
3. On input small $k \in R_q$, the server sends $\boldsymbol{d}_x = \boldsymbol{c}_x \cdot k + \boldsymbol{e}'$ for small $\boldsymbol{e}' \in R_q^{1\times\ell}$.
4. The client outputs $\boldsymbol{y} = \left\lfloor \frac{p}{q} \cdot (\boldsymbol{d}_x - \boldsymbol{c} \cdot s) \right\rceil$.

For server security, note that $\boldsymbol{d}_x = \boldsymbol{a}\cdot s\cdot k + \boldsymbol{a}^F(x)\cdot k + \boldsymbol{e}_1\cdot k + \boldsymbol{e}'$. Suppose that we choose $\boldsymbol{e}'$ from a distribution that hides addition of terms $\boldsymbol{e}_1 \cdot k, \boldsymbol{e}\cdot s$ and $\boldsymbol{e}_x$ (where $\boldsymbol{e}_x$ is from some other narrow distribution). Then, from the perspective of the client, the server might as well have sent $\boldsymbol{d}_x = (\boldsymbol{a}\cdot k + \boldsymbol{e})\cdot s + \boldsymbol{e}' + (\boldsymbol{a}^F(x)\cdot k + \boldsymbol{e}_x) = \boldsymbol{c}\cdot s + (\boldsymbol{a}^F(x)\cdot k + \boldsymbol{e}_x) + \boldsymbol{e}'$. Picking $\boldsymbol{e}_x$ from an appropriate distribution [5] makes the term in brackets i.e. $\boldsymbol{a}^F(x) \cdot k + \boldsymbol{e}_x$ computationally indistinguishable from uniform random under a RLWE assumption, even given the value of $\boldsymbol{c}$ which is also indistinguishable from random by a RLWE assumption. This implies that the message $\boldsymbol{d}_x$ leaks nothing about the server's key k.

For client security, we pick s from a valid RLWE secret distribution and a Gaussian $\boldsymbol{e}$. This implies that $\boldsymbol{c}_x = \boldsymbol{a} \cdot s + \boldsymbol{e} + \boldsymbol{a}^F(x)$ is indistinguishable from uniform by RLWE. Finally, we must show that the client does indeed recover $F_k(x)$ as its output $\boldsymbol{y}$. For correctness, we *would like to say* that

$$\left\lfloor \frac{p}{q} \cdot (\boldsymbol{d}_x - \boldsymbol{c} \cdot s) \right\rceil = \left\lfloor \frac{p}{q} \cdot \boldsymbol{a}^F(x) \cdot k + \frac{p}{q}(\boldsymbol{e}_1 \cdot k - \boldsymbol{e} \cdot s + \boldsymbol{e}') \right\rceil = \left\lfloor \frac{p}{q} \cdot \boldsymbol{a}^F(x) \cdot k \right\rceil.$$

Thus, we guarantee correctness if all coefficients of $\frac{p}{q} \cdot \boldsymbol{a}^F(x) \cdot k$ are at least $\left| \frac{p}{q} (\boldsymbol{e}_1 \cdot k - \boldsymbol{e} \cdot s + \boldsymbol{e}') \right|_\infty$ away from $\mathbb{Z} + \frac{1}{2}$. It turns out that this condition is

satisfied with extremely high probability due to the 1-dimensional short integer solution (1D-SIS) assumption [15] regardless of the way an efficient server chooses its key. The form of $\boldsymbol{a}^F(x)$ is crucial to the connection with the 1D-SIS problem. In particular, we rely on the fact that we can decompose $\boldsymbol{a}^F(x)$ as $\boldsymbol{a}'_1 \cdot \boldsymbol{a}'_2$ where $\boldsymbol{a}'_1 \in R_q^{1\times\ell}$ is uniform random and $\boldsymbol{a}'_2 \in R_q^{\ell\times\ell}$ has entries that are polynomials with *binary* coefficients.

Ultimately, the security of our VOPRF construction (with particular choices of NIZKAoK instantiations) holds in the QROM and relies on the hardness of sub-exponential RLWE and 1D-SIS which are both at least as hard as certain lattice problems. We discuss parameters in Section 5.3.

Related Work & Discussion. A related primitive to a VOPRF is a verifiable random function (VRF). A VRF is a keyed pseudorandom function allowing an entity with the key to create publicly verifiable proofs of correct evaluation. Recently, Yang et al. [58] showed a lattice-based construction of a VRF based using the definition of [47]. In fact, the proof systems of Yang et al. serve as a crucial foundation for one way of instantiating the proof systems used in our VOPRF. However, it should be noted that the Yang et al. construction (like ours) is not in the standard model due to the use of the Fiat-Shamir [26] transform.

While our work provides a first construction for a post-quantum VOPRF, it does not resolve this question completely. The reason VOPRFs enjoy popularity is their efficiency in the discrete logarithm setting. In contrast, our construction – while practically instantiable – is far less efficient. This relative inefficiency is partly due to our choice of relying on lattice-based constructions for our zero-knowledge proof systems, along with the super-polynomial factors required for the RLWE-based PRF and noise drowning. Improving these areas thus suggests ways to achieve concretely more efficient schemes. In fact, we do discuss attempts to optimise our main protocol with a view to reducing the impact of the zero-knowledge proofs. In particular, one can amortise the costs of the client zero-knowledge proof by sending queries in batches and sending one proof of a more complex statement. This saves a small additive term in the overall cost compared to sending the queries one at a time. Additionally, we discuss the use of a cut-and-choose approach to removing the server's zero-knowledge proof at the effective cost of extra repetitions of the protocol. Ultimately, this does not improve overall efficiency, but it does dramatically reduce the burden on the server. For more details, see Section 3.2. An alternative approach is to accept, for now, that VOPRFs are less appealing building blocks in a post-quantum world, and to revisit their applications to provide post-quantum alternatives on a per application basis.

One could alternatively instantiate VOPRFs using generic techniques for establishing Multi-Party Computation (MPC) protocols by treating a single execution of the VOPRF protocol, for a PRF like AES, as a single invocation of a classical two-party *actively secure* MPC protocol. But this does not give the round-optimality that we are after. See Appendix E for a discussion about this.

A previous draft of this work presented a protocol roughly analogous to the main one studied in this work. To give intuition, the starting point of our previous

protocol involved the client hiding $\boldsymbol{a}^F(x)$ by *multiplying* by its own secret $s \in R_q$ and adding some error. The client had to multiply by $s^{-1} \in R_q$ (assuming it exists) to complete the protocol and it turned out that both s and s^{-1} both had to be small for correctness. However, for Gaussian s, this is simply not the case. In order to overcome this obstacle, we used a method for sampling "full" NTRU keys [32,52]. In more detail, we use the fact that for Gaussian $s, t \in R_q$, we can efficiently find short (u, v) such that $u \cdot s + v \cdot t = 1$. Although this trick is not necessary for building a VOPRF,[9] we include it in Appendix D as we believe that it may have applications elsewhere.

Road Map. We begin with preliminaries in Section 2. Note that Definition 1 deviates from the usual MPC definition. In particular, we argue security against malicious clients when k is sampled from a key distribution for which the PRF is pseudorandom, rather than arguing security for arbitrary fixed k. Next is the VOPRF construction and discussion of optimisations (Section 3) followed by a high-level description of the zero-knowledge proof instantiations (Section 4). Finally, we give the security proof for our VOPRF protocol in Section 5.

Our appendices consist of a more detailed account of our computational hardness assumptions (Appendix A) followed by a collection of miscellaneous results (Appendix B) and more details of our zero-knowledge instantiations (Appendix C). To supplement the main body of this work, we also discuss an alternative construction along with a trick (Appendix D), a generic MPC construction (Appendix E), and a proof-of-concept implementation in Sage [56] ignoring zero-knowledge proofs and distributions (Appendix F).

2 Preliminaries

All algorithms will be considered to be randomised algorithms unless explicitly stated otherwise. A PPT algorithm is a randomised (i.e. probabilistic) algorithm with polynomial running time in the security parameter κ. We consider the probability distribution of outputs of algorithms as being over all possible choices of the internal coins of the algorithm. For a distribution $\mathcal{D}$, we denote the sampling of x according to distribution $\mathcal{D}$ by $x \leftarrow \mathcal{D}$. We write $x \leftarrow S$ for a finite set S to indicate sampling uniformly at random from S. We use the notation $\mathcal{D}_1 \approx_c \mathcal{D}_2$ to mean the distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ are computationally indistinguishable and $\approx_s$ to denote statistical indistinguishability. We use the standard asymptotic notations. We let $\mathsf{negl}(\kappa)$ denote a negligible function (i.e. a function that is $\kappa^{-\omega(1)}$) and write $r_1 \gg r_2$ as short-hand for $r_1 \geq \kappa^{\omega(1)} \cdot r_2$. We say a distribution $\mathcal{D}$ is (B, δ)-bounded if $\Pr[\|x\| \geq B \mid x \leftarrow \mathcal{D}] < \delta$. If a distribution is (B, δ)-bounded for a negligible δ, then we say that distribution is simply B-bounded.

In this work we will use power of two cyclotomic rings. In particular, for some integer q, we will be considering polynomials in the power-of-two cyclotomic ring $R = \mathbb{Z}[X]/\langle X^n + 1\rangle$ and $R_q := R/qR$ where n is a power-of-two. $R_{\leq c}$ is the set of elements of R where all coefficients have an absolute value at most c. We also

[9] as pointed out by a previous reviewer

use a rounding operation from $\mathbb{Z}_q$ to $\mathbb{Z}_{q'}$ where $q' < q$. For $x \in \mathbb{Z}_q$, this rounding operation is defined as

$$\lfloor x \rceil_{q'} := \lfloor (q'/q) \cdot x \rceil$$

where $\lfloor \cdot \rceil$ denotes rounding to the nearest integer (rounding down in the case of a tie). If q' divides q, we can lift rounded integers back up to $\mathbb{Z}_q$ by simply multiplying by q/q'. Note that lifting the result of a rounding takes an $x \in \mathbb{Z}_q$ to the nearest multiple of q/q'. Therefore, the difference between x and the result of this rounding then lifting is at most $q/(2 \cdot q')$. Polynomials and vectors are rounded component-wise. We write $\|\cdot\|$ for the Euclidean norm and $\|\cdot\|_\infty$ for the infinity norm. We define the norms of ring elements by considering the norms of their *coefficient* vectors. Vectors whose entries are ring elements will be denoted using bold characters and integer vectors will be indicated by an over-arrow e.g. $\boldsymbol{v}$ has ring entries and $\vec{w}$ has integer entries. Suppose $\boldsymbol{v} = (v_1, \ldots, v_n)$. A norm of $\boldsymbol{v}$ is the norm of the vector obtained by concatenating the coefficient vectors of $v_1, \ldots, v_n$.

Gaussian distributions. For any $\sigma > 0$, define the *Gaussian function* on $\mathbb{R}^n$ centred at $\boldsymbol{c} \in \mathbb{R}^n$ with parameter σ to be:

$$\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x}) = e^{-\pi \cdot \|\boldsymbol{x}-\boldsymbol{c}\|^2/\sigma^2}, \ \forall\, \boldsymbol{x} \in \mathbb{R}^n.$$

Define $\rho_\sigma(\mathbb{Z}) := \sum_{i \in \mathbb{Z}} \rho_\sigma(i)$. The *discrete Gaussian distribution over* $\mathbb{Z}$, denoted χ_σ assigns probability $\rho_\sigma(i)/\rho_\sigma(\mathbb{Z})$ to each $i \in \mathbb{Z}$ and probability 0 to each non-integer point. The *discrete Gaussian distribution over* R, denoted as $R(\chi_\sigma)$, is the distribution over R where each coefficient is distributed according to χ_σ. Using the results of [28,13], χ_σ can be sampled in polynomial time. Moreover the Euclidean norm of a sample from $R(\chi_\sigma)$ can be bounded using an instantiation of Lemma 1.5 of [4]. We state this lemma next.

Lemma 1. *Let $\sigma > 0$ and $n = \mathsf{poly}(\kappa)$. Then*

$$\Pr\left[\|x\| \geq \sigma\sqrt{n} \;\middle|\; x \leftarrow R(\chi_\sigma)\right] < \mathsf{negl}(\kappa)\,.$$

In addition, following the same reasoning as in [24] we have the following "drowning/smudging" lemma.

Lemma 2. *Let $\sigma > 0$ and $y \in \mathbb{Z}$. The statistical distance between χ_σ and $\chi_\sigma + y$ is at most $|y|/\sigma$.*

2.1 Verifiable Oblivious Pseudorandom Functions

Recall that the main goal of our work is to build a verifiable oblivious pseudorandom function (VOPRF). A VOPRF is a protocol between two parties: a server $\mathbb{S}$ and a client $\mathbb{C}$, securely realising the ideal functionality in Figure 1. The functionality consists of two phases, the initialisation phase and the query phase. The initialisation phase is divided into two steps: one run once by the server, and one run once by any client who wishes to utilise the VOPRF provided by

the server. In the event that the functionality $\mathcal{F}_{\mathsf{VOPRF}}$ receives a valid input k from $\mathbb{S}$ during the initialisation phase, it stores the key for use during the query phase. This models a server ($\mathbb{S}$) in a real protocol committing to a PRF key k.

Next comes the query phase, where a client $\mathbb{C}$ sends some value x to $\mathcal{F}_{\mathsf{VOPRF}}$. Once this value x has been received, the server $\mathbb{S}$ either sends the functionality an instruction to abort or to deliver the value $y = F_k(x)$ to $\mathbb{C}$. Finally, the functionality carries out this instruction. Importantly, (assuming that no abort is triggered) the client has the guarantee that its output is indeed $F_k(x)$ i.e. the output of the client is *verifiably* correct when interacting with $\mathcal{F}_{\mathsf{VOPRF}}$.

This is a two party functionality between a server $\mathbb{S}$ and a client $\mathbb{C}$. We assume there is a fixed PRF function defined by $F_k(x)$.

Init-S: On input of init from the server the functionality waits for an input k from party $\mathbb{S}$. If $\mathbb{S}$ returns abort then the functionality aborts. Otherwise, the functionality stores the value k if it is a *valid key*† and aborts if not.

Init-C: On input of init from a client, the functionality will return abort if the init procedure for the server has not successfully completed.

Query: On input of (query, x) from a client $\mathbb{C}$, if $x \neq \perp$ then the functionality waits for an input from party $\mathbb{S}$. If $\mathbb{S}$ returns deliver then the functionality sends $y = F_k(x)$ to party $\mathbb{C}$. If $\mathbb{S}$ returns abort then the functionality aborts.

Figure 1. The Ideal Functionality $\mathcal{F}_{\mathsf{VOPRF}}$. †The notion of a valid key refers to whether the key conforms to a pre-determined distribution. See Definition 1 for more details on this requirement.

We now describe the distributions that arise in the security requirement. We consider malicious adversaries throughout that behave arbitrarily and begin with the distributions of interest when a server has been corrupted. First, we consider a "real" world protocol Π between $\mathbb{C}(x)$ and $\mathbb{S}(k)$ along with an adversary $\mathcal{A}$. We denote $\mathsf{real}_{\Pi,\mathcal{A},\mathbb{S}}(x, k, 1^\kappa)$ to be the joint output distribution of $\mathcal{A}(k)$ (when corrupting $\mathbb{S}(k)$) and $\mathbb{C}(x)$ where $\mathbb{C}(x)$ behaves as specified by Π. In this setting, $\mathcal{A}$ interacts directly with $\mathbb{C}$. Now we introduce a simulator denoted Sim that lives in the "ideal" world. Specifically, still assuming $\mathcal{A}$ corrupts a server, Sim interacts with $\mathcal{A}$ on one hand and with $\mathbb{C}(x)$ via $\mathcal{F}_{\mathsf{VOPRF}}$ on the other. In this setting, for any client/server input pair (x, k), we define $\mathsf{ideal}_{\mathcal{F}_{\mathsf{VOPRF}},\mathsf{Sim},\mathcal{A},\mathbb{S}}(x, k, 1^\kappa)$ to be the joint output distribution of $\mathcal{A}(k)$ and the honest client $\mathbb{C}(x)$ when $\mathcal{A}(k)$ interacts via Sim. Informally, one may interpret Sim as an attacker-in-the-middle between $\mathcal{A}$ and the outside world where Sim interacts with $\mathcal{F}_{\mathsf{VOPRF}}$ external to the view of $\mathcal{A}$. Security argues that whatever $\mathcal{A}$ can learn/affect in the real protocol can be emulated via Sim in the ideal setting.

Next, we describe the distributions of interest when a client has been corrupted by an adversary $\mathcal{A}$. We let $\mathcal{K}$ denote the key distribution under which PRF security of F holds. First, consider a "real" world case where $\mathcal{A}$ corrupts $\mathbb{C}(x)$ and directly interacts with honest $\mathbb{S}(k)$ which follows the specification of

protocol Π. In this case, we use $\mathsf{real}_{\Pi,\mathcal{A},\mathbb{C}}(x,\mathcal{K},1^\kappa)$ to denote the joint output distribution of $\mathcal{A}(x)$ and $\mathbb{S}(k)$[10] where $k \leftarrow \mathcal{K}$. Now consider an alternative "ideal" world case where we introduce a simulator Sim interacting with $\mathcal{A}$ on one hand and with $\mathbb{S}(x)$ via $\mathcal{F}_{\mathsf{VOPRF}}$ on the other hand. Once again, one may wish to interpret the simulator as an attacker-in-the-middle interacting with $\mathcal{F}_{\mathsf{VOPRF}}$ external to the view of $\mathcal{A}$. In this alternative case, we denote the joint output distribution of $\mathcal{A}(x)$ and $\mathbb{S}(k)$ where $\mathcal{A}$ interacts via Sim and $k \leftarrow \mathcal{K}$ as $\mathsf{ideal}_{\mathcal{F}_{\mathsf{VOPRF}},\mathsf{Sim},\mathcal{A},\mathbb{C}}(x,\mathcal{K},1^\kappa)$.

Finally, for protocol Π, let $\mathsf{output}(\Pi,x,k)$ denote the output distribution of a *client* with input x running protocol Π with a server whose input key is k. Using the notation established above, we can present our definition of a VOPRF.

Definition 1. *A protocol Π is a verifiable oblivious pseudorandom function if all of the following hold:*

1. ***Correctness:*** *For every pair of inputs (x,k),*

$$\Pr[\mathsf{output}(\Pi,x,k) \neq F_k(x)] \leq \mathsf{negl}(\kappa)\,.$$

2. ***Malicious server security:*** *For any PPT adversary $\mathcal{A}$ corrupting a server, there exists a PPT simulator Sim such that for every pair of inputs (x,k):*

$$\mathsf{ideal}_{\mathcal{F}_{\mathsf{VOPRF}},\mathsf{Sim},\mathcal{A},\mathbb{S}}(x,k,1^\kappa) \approx_c \mathsf{real}_{\Pi,\mathcal{A},\mathbb{S}}(x,k,1^\kappa).$$

3. ***Average case malicious client security:*** *For any PPT adversary $\mathcal{A}$ corrupting a client, there exists a PPT simulator Sim such that for all client inputs x:*
   - $\mathsf{ideal}_{\mathcal{F}_{\mathsf{VOPRF}},\mathsf{Sim},\mathcal{A},\mathbb{C}}(x,\mathcal{K},1^\kappa) \approx_c \mathsf{real}_{\Pi,\mathcal{A},\mathbb{C}}(x,\mathcal{K},1^\kappa)$.
   - *If $\mathcal{A}$ correctly outputs $F_k(x)$ with all but negligible probability over the choice $k \leftarrow \mathcal{K}$ when interacting directly with $\mathbb{S}(k)$ using protocol Π, then $\mathcal{A}$ also outputs $F_k(x)$ with all but negligible probability when interacting via Sim.*

We now discuss this definition. Note that the correctness and malicious server security requirements are the standard ones used in MPC. Therefore, we restrict this discussion to the condition that we call average case malicious client security. The motivation for this non-standard property is that an honest server will always sample a key from distribution $\mathcal{K}$ as it wishes to provide *pseudorandom* function evaluations. In particular, PRF security holds with respect to this key distribution $\mathcal{K}$. Therefore, it makes sense to ask what a malicious client may learn/affect only in the case where $k \leftarrow \mathcal{K}$ which leads to the first point of our average case malicious client security requirement. The second point of the requirement captures the fact that adversaries may have access to an oracle that checks whether the PRF was evaluated correctly or not. Suppose that we give the adversary $\mathcal{A}$ access to an oracle which can check an input/output pair to the PRF is valid or not. Then $\mathcal{A}$ should not be able to distinguish whether

[10] Note that the output of $\mathbb{S}(k)$ is $\perp$ in our construction.

it is interacting with a real server $\mathbb{S}$ or a simulation Sim. Note that our proof structure relies heavily on our alternative malicious client security definition. In particular, the definition above allows us to argue over the entropy of secret keys when making indistinguishability claims.

Alternative definitions Note that alternative security definitions exist for (V)OPRFs. In the UC security models that are favoured by Jarecki et al. [33,34] the output of the PRF is wrapped in the output of a programmable random oracle evaluation. This is a fact that is utilised by the OPAQUE PAKE protocol [35] that allows arguing that the pseudorandom function evaluations are pseudorandom *even* to the server (the key-holder). Unfortunately, using a similar technique here is difficult as constructing programmable random oracles in the quantum random oracle model (QROM) is known to be difficult [10].

2.2 Computational Assumptions

Here we present the presumed quantum hard computational problems that will be used in our security proofs. Evidence that these problems are indeed quantum hard follows via reductions from standard lattice problems (see Appendix A). These reductions from lattice problems will be used to asymptotically analyse secure parameter settings for our VOPRF. The first is the standard decisional RLWE problem [45].

Definition 2. (RLWE problem) *Let $q, m, n, \sigma > 0$ depend on κ (q, m, n are integers). The* decision-RLWE problem ($\mathsf{dRLWE}_{q,n,m,\sigma}$) *is to distinguish between:*

$$(a_i,\ a_i \cdot s + e_i)_{i\in[m]} \in (R_q)^2 \quad \text{and} \quad (a_i, u_i)_{i\in[m]} \in (R_q)^2$$

for $a_i, u_i \leftarrow R_q$; $s, e_i \leftarrow R(\chi_\sigma)$.

We sometimes write $\mathsf{dRLWE}_{q,n,\sigma}$, leaving the parameter m (representing the number of samples) implicit. The second problem is slightly less standard. It is the short integer solution problem in *dimension 1* (1D-SIS). The following formulation of the problem was used in [15] in conjunction with a lemma attesting to its hardness. See Appendix A for more details.

Definition 3. (1D-SIS, [15, Definition 3.4]) *Let q, m, t depend on κ. The one-dimensional SIS problem, denoted $\mathsf{1D\text{-}SIS}_{q,m,t}$, is the following: Given a uniform $\boldsymbol{v} \leftarrow \mathbb{Z}_q^m$, find $\boldsymbol{z} \in \mathbb{Z}^m$ such that $||\boldsymbol{z}||_\infty \leq t$ and $\langle \boldsymbol{v}, \boldsymbol{z} \rangle \in [-t, t] + q\mathbb{Z}$.*

2.3 Non-Interactive Zero-Knowledge Arguments of Knowledge

The foundations of zero-knowledge (ZK) proof systems were established in a number of works [26,31,30,9]. At a high level, a ZK proof system for language $\mathcal{L}$ allows a prover $\mathbb{P}$ to convince a verifier $\mathbb{V}$ that some instance x is in $\mathcal{L}$, without revealing anything beyond this statement. Further, a ZK argument of knowledge (ZKAoK) system allows $\mathbb{P}$ to convince $\mathbb{V}$ that they hold a witness w attesting to the fact that x is in $\mathcal{L}$ (where the $\mathcal{L}$ is defined by a relation predicate $\mathsf{P}_\mathcal{L}(x, w)$).

Definition 4. (NIZKAoK) *Let* $\mathbb{P}$ *be a* prover*, let* $\mathbb{V}$ *be a* verifier*, let* $\mathcal{L}$ *be a language with accompanying relation predicate* $\mathsf{P}_{\mathcal{L}}(\cdot,\cdot)$*. Let* $\mathcal{W}_{\mathcal{L}}(x)$ *be a generic set of witnesses attesting to the fact that* $x \in \mathcal{L}$*, i.e.* $\forall x \in \mathcal{L}$*, and* $w \in \mathcal{W}_{\mathcal{L}}(x)$ *we have* $\mathsf{P}_{\mathcal{L}}(x,w) = 1$*. Let* $\mathsf{nizk} = (\mathsf{Setup}, \mathbb{P}, \mathbb{V})$ *be a tuple of algorithms defined as follows:*

- $\mathsf{crs} \leftarrow \mathsf{nizk.Setup}(1^\kappa)$*: outputs a common random string* crs*.*
- $\pi \leftarrow \mathsf{nizk}.\mathbb{P}(\mathsf{crs}, x, w)$*: on input* crs*, a word* $x \in \mathcal{L}$ *and a witness* $w \in \mathcal{W}_{\mathcal{L}}(x)$*; outputs a proof* $\pi \in \{0,1\}^{\mathsf{poly}(\kappa)}$*.*
- $b \leftarrow \mathsf{nizk}.\mathbb{V}(\mathsf{crs}, x, \pi)$*: on input* crs*, a word* $x \in \mathcal{L}$ *and a proof* $\pi \in \{0,1\}^{\mathsf{poly}(\kappa)}$*; outputs* $b \in \{0,1\}$*.*

Definition 5. (NIZKAoK Security) *We say that* nizk *is a* non-interactive zero-knowledge argument of knowledge *(NIZKAoK) for* $\mathcal{L}$ *if the following holds.*

1. (Completeness)*: Consider* $x \in \mathcal{L}$ *and* $w \in \mathcal{W}_{\mathcal{L}}(x)$*, where* $\mathsf{P}_{\mathcal{L}}(x,w) = 1$*. Then:*

$$\Pr\left[1 \leftarrow \mathsf{nizk}.\mathbb{V}(\mathsf{crs}, x, \pi) \middle| \begin{smallmatrix}\mathsf{crs}\leftarrow\mathsf{nizk.Setup}(1^\kappa)\\ \pi\leftarrow\mathsf{nizk}.\mathbb{P}(\mathsf{crs},x,w)\end{smallmatrix}\right] \geq 1 - \mathsf{negl}(\kappa)\,.$$

2. (Computational knowledge extraction)*: The proof system satisfies computational knowledge extraction with knowledge error* $\bar{\kappa}$ *if, for any PPT prover* $\mathbb{P}^*$ *with auxiliary information* aux*, the following holds. There exists a PPT algorithm* $\mathsf{nizk.Extract}$ *and a polynomial* p *such that, for any input* x*, then:*

$$\Pr[1 \leftarrow \mathsf{P}_{\mathcal{L}}(x,w') | w' \leftarrow \mathsf{nizk.Extract}(\mathbb{P}^*(\mathsf{crs}, x, \mathsf{aux}))] \geq \frac{\nu - \bar{\kappa}}{p(|x|)}$$

   is satisfied, where ν *is the probability that* $\mathsf{nizk}.\mathbb{V}(\mathsf{crs}, x, \mathbb{P}^*(\mathsf{crs}, x, \mathsf{aux}))$ *outputs 1.*
3. (Computational zero-knowledge)*: There exists a simulated setup algorithm* $\mathsf{nizk.SimSetup}(1^\kappa)$ *outputting* $\mathsf{crs}_{\mathsf{Sim}}$ *and a trapdoor* $\mathcal{T}$ *along with a PPT algorithm* $\mathsf{nizk.Sim}(\mathsf{crs}_{\mathsf{Sim}}, \mathcal{T}, x)$ *satisfying*

$$\left\{\begin{smallmatrix}\mathsf{crs}\leftarrow\mathsf{nizk.Setup}(1^\kappa)\\ \pi\leftarrow\mathsf{nizk}.\mathbb{P}(\mathsf{crs},x,w)\end{smallmatrix}\right\} \approx_c \left\{\pi_{\mathsf{Sim}} \leftarrow^{\mathsf{crs}_{\mathsf{Sim}}}_{\mathsf{nizk.Sim}(\mathsf{crs}_{\mathsf{Sim}},\mathcal{T},x)} \middle| (\mathsf{crs}_{\mathsf{Sim}}, \mathcal{T}) \leftarrow \mathsf{nizk.SimSetup}(1^\kappa)\right\}$$

   $\forall x \in \mathcal{L}$ *and* $w \in \mathcal{W}_{\mathcal{L}}(x)$*.*

Interactive proof systems. An interactive proof system is one where the proving algorithm ($\mathbb{P}$) requires interaction with the verifier. Such an interaction could be an arbitrary protocol, with many message exchanges, but a typical (in the honest verifier case) scenario is a three-move protocol consisting of a commitment (from the prover), a uniformly chosen challenge (from the verifier) and then a response (from the prover). Such protocols are referred to as Σ-protocols. Fiat and Shamir [26] established a mechanism of switching a (constant-round) honest verifier zero-knowledge interactive proof of knowledge into a *non-interactive* zero-knowledge proof of knowledge in the random oracle model (ROM). In particular, the random challenge provided by the verifier is replaced with the output

of a random oracle evaluation taking as input the statement x and the provers initial commitment. It was recently shown that the standard Fiat-Shamir transform is also secure in the *quantum* ROM (QROM) [25,42] assuming the underlying Σ-protocol satisfies certain properties.

2.4 Lattice PRF

We will use an instantiation of the lattice PRF from [5]. Below, we present relevant definitions/results, all of which are particular cases of definitions/results from [5]. We set $\ell = \lceil \log_2 q \rceil$ throughout. The construction from [5] makes use of *gadget matrices* that can be found in many previous works [48,5,15,29].

Gadgets G, G^{-1} Define $G : R_q^{\ell \times \ell} \to R_q^{1 \times \ell}$ to be the linear operation corresponding to left multiplication by $(1, 2, \ldots, 2^{\ell-1})$. Further, define $G^{-1} : R_q^{1 \times \ell} \to R_q^{\ell \times \ell}$ to be the bit decomposition operation that essentially inverts G i.e. the i^{th} column of $G^{-1}(\boldsymbol{a})$ is the bit decomposition of $a_i \in R_q$ into binary polynomials.

The instantiation of [5] that we will present our VOPRF with respect to is defined as $F_k(x) = \lfloor \boldsymbol{a}_x \cdot k \rceil_p$ for $\boldsymbol{a}_x \in R_q^{1 \times \ell}$ given below.

Definition 6. *Fix some* $\boldsymbol{a}_0, \boldsymbol{a}_1 \leftarrow R_q^{1 \times \ell}$. *For any* $x = (x_1, \ldots, x_L) \in \{0,1\}^L$. *We define* $\boldsymbol{a}_x \in R_q^{1 \times \ell}$ *as*

$$\boldsymbol{a}_x := \boldsymbol{a}_{x_1} \cdot G^{-1}\left(\boldsymbol{a}_{x_2} \cdot G^{-1}\left(\boldsymbol{a}_{x_3} \cdot G^{-1}\left(\ldots\left(\boldsymbol{a}_{x_{L-1}} \cdot G^{-1}\left(\boldsymbol{a}_{x_L}\right)\right)\right)\right)\right) \in R_q^{1 \times \ell}.$$

The pseudorandomness of this construction follows from the ring learning with errors (RLWE) assumption (with normal form secrets).

Theorem 1 ([5]). *Sample* $k \leftarrow R(\chi_\sigma)$. *If* $q \gg p \cdot \sigma \cdot \sqrt{L} \cdot n \cdot \ell$, *then the function* $F_k(x) = \lfloor \boldsymbol{a}_x \cdot k \rceil_p$ *is a PRF under the* $\mathsf{dRLWE}_{q,n,\sigma}$ *assumption.*

When we eventually prove security of our VOPRF, it will be useful to define a special error distribution such that $\boldsymbol{a}_x \cdot k + \boldsymbol{e}$ remains indistinguishable from uniform (under RLWE) when $\boldsymbol{e}$ is sampled from this special error distribution. To this end, we introduce the distributions $\mathcal{E}_{\boldsymbol{a}_0, \boldsymbol{a}_1, x, \sigma}$ followed by a lemma that is implicit in the pseudorandomness proof of the PRF from [5].

Definition 7. *For* $\boldsymbol{a}_0, \boldsymbol{a}_1 \in R_q^{1 \times \ell}$, *define*

$$\boldsymbol{a}_{x \setminus i} := G^{-1}\left(\boldsymbol{a}_{x_{i+1}} \cdot G^{-1}\left(\boldsymbol{a}_{x_{i+2}} \cdot G^{-1}\left(\cdots\left(\boldsymbol{a}_{x_{L-1}} \cdot G^{-1}\left(\boldsymbol{a}_{x_L}\right)\right)\cdots\right)\right)\right) \in R_q^{\ell \times \ell}.$$

Furthermore, let $\mathcal{E}_{\boldsymbol{a}_0, \boldsymbol{a}_1, x, \sigma}$ *be the distribution that is sampled by choosing* $\boldsymbol{e}_i \leftarrow R(\chi_\sigma)^{1 \times \ell}$ *for* $i = 1, \ldots, L$ *and outputting*

$$\boldsymbol{e} = \sum_{i=1}^{L-1} \boldsymbol{e}_i \cdot \boldsymbol{a}_{x \setminus i} + \boldsymbol{e}_L.$$

Lemma 3 (Implicit in [5]). *If* $\boldsymbol{a}_0, \boldsymbol{a}_1 \leftarrow R_q^{1\times\ell}, \boldsymbol{e} \leftarrow \mathcal{E}_{\boldsymbol{a}_0,\boldsymbol{a}_1,x,\sigma}$ *and* $s \leftarrow R(\chi_\sigma)$*, then for any fixed* $x \in \{0,1\}^L$*,*

$$(\boldsymbol{a}_0,\ \boldsymbol{a}_1,\ \boldsymbol{a}_x \cdot s + \boldsymbol{e})$$

is indistinguishable from uniform random by the $\mathsf{dRLWE}_{q,n,\sigma}$ *assumption.*

In addition to introducing $\mathcal{E}_{\boldsymbol{a}_0,\boldsymbol{a}_1,x,\sigma}$, it will be useful to write down an upper bound on the infinity norm on errors drawn from this distribution. The following lemma follows from the fact that for $y \leftarrow \chi_\sigma$, $\|y\|_\infty \leq \sigma\sqrt{n}$ with all but negligible probability by Lemma 1. In fact, we could use the result that $\|y\|_\infty \leq \sigma n^{c'}$ with probability at least $1 - c \cdot \exp(-\pi n^{2c'})$ for any constant $c' > 0$ and some universal constant c to reduce the upper bound, but we choose not to for simplicity.

Lemma 4 (Bound on errors). *Let* $x \in \{0,1\}^L$*,* $\ell = \lceil \log_2 q \rceil$ *and* $n = \mathsf{poly}(\kappa)$*. Samples from* $\mathcal{E}_{\boldsymbol{a}_0,\boldsymbol{a}_1,x,\sigma}$ *have infinity norm at most* $L \cdot \ell \cdot \sigma \cdot n^{3/2}$ *with all but negligible probability.*

3 A VOPRF Construction from Lattices

In this section, we provide a construction emulating the DH blinding construction $H(x)^k = (H(x) \cdot g^r)^k / (g^k)^r$. In what follows, we will initially ignore the zero-knowledge proofs establishing that all computations are performed honestly. A detailed description of the protocol is in Figure 2 but the main high-level idea follows.

Recall that we are working with power-of-two cyclotomic rings. Informally, suppose a *client* wants to obtain $a' \cdot k + e' \in R_q$ (where e' is relatively small) from a server holding a *short* k without revealing $a' \in R_q$. Further, suppose that the server has published an LWE instance $(a, c := a \cdot k + e)$ for truly uniformly a and small Gaussian e. One way to achieve our goal is to have the client compute $c_x := a \cdot s + e_1 + a'$ for Gaussian (s, e_1). Next the server responds by computing $d := c_x \cdot k + e''$ for relatively small e'' and the client finally outputs

$$\begin{aligned} d - c \cdot s &= (a \cdot s + e_1 + a') \cdot k + e'' - (a \cdot k + e) \cdot s \\ &= a' \cdot k + (e_1 \cdot k + e \cdot s + e'') \\ &\approx a' \cdot k. \end{aligned}$$

The above gives the intuition behind our actual protocol. Roughly, the idea is to replace a' with $\boldsymbol{a}_x$ from a BP14 evaluation. As mentioned above, a more detailed formulation of our construction is given in Figure 2. In the protocol description, $\mathbb{P}_i$ and $\mathbb{V}_i$ denote prover and verifier algorithms for three different zero-knowledge argument systems indexed by $i \in \{0,1,2\}$.

CRS SetUp: To set up the CRS execute the following steps:

- Pick $\boldsymbol{a}_0, \boldsymbol{a}_1 \leftarrow R_q^{1\times\ell}$
- Sample $\boldsymbol{a} \leftarrow R_q^{1\times\ell}$, sample $\overline{\mathsf{crs}}_0$ for proof system $\mathbb{P}_0$ and set $\mathsf{crs}_0 := (\overline{\mathsf{crs}}_0, \boldsymbol{a})$
- Sample crs_1 and crs_2 for proof systems $\mathbb{P}_1$ and $\mathbb{P}_2$ respectively

Init: The initialisation procedure is executed by the server $\mathbb{S}$ and a client $\mathbb{C}$ both with initial input crs_0.

- **Init-S:** The server $\mathbb{S}$ executes the following steps
  - $k \leftarrow R(\chi_\sigma), \boldsymbol{e} \leftarrow R(\chi_\sigma)^{1\times\ell}$.
  - $\boldsymbol{c} \leftarrow \boldsymbol{a} \cdot k + \boldsymbol{e} \bmod q$.
  - $\pi_0 \leftarrow \mathbb{P}_0(k, e : \mathsf{crs}_0, \boldsymbol{c})$.

  and sends $(\boldsymbol{c}, \pi_0)$ to a client $\mathbb{C}$.
- **Init-C:** On receipt of $(\boldsymbol{c}, \pi_0)$ a client executes
  - $b \leftarrow \mathbb{V}_0(\mathsf{crs}_0, \boldsymbol{c}, \pi_0)$.
  - Output abort if $b = 0$, otherwise store $\boldsymbol{c}$.

Query: This is a two round protocol between a client and the server, with a client going first.

1. On input of $(x \in \{0,1\}^L, \mathsf{crs}_1, \mathsf{crs}_2)$ a client $\mathbb{C}$ executes the following steps
   - $s \leftarrow R(\chi_\sigma), \boldsymbol{e}_1 \leftarrow R(\chi_\sigma)^{1\times\ell}$.
   - $\boldsymbol{a}_x = \boldsymbol{a}_{x_1} \cdot G^{-1}\left(\cdots\left(\boldsymbol{a}_{x_{L-1}} \cdot G^{-1}\left(\boldsymbol{a}_{x_L}\right)\right)\cdots\right) \bmod q$.
   - $\boldsymbol{c}_x \leftarrow \boldsymbol{a} \cdot s + \boldsymbol{e}_1 + \boldsymbol{a}_x \bmod q$.
   - $\pi_1 \leftarrow \mathbb{P}_1(x, s, \boldsymbol{e}_1 : \mathsf{crs}_1, \boldsymbol{c}_x, \boldsymbol{a}, \boldsymbol{a}_0, \boldsymbol{a}_1)$.

   and sends $(\boldsymbol{c}_x, \pi_1)$ to the server $\mathbb{S}$.
2. On receipt of $(\boldsymbol{c}_x, \pi_1)$ the server $\mathbb{S}$ executes the following steps
   - $b \leftarrow \mathbb{V}_1(\mathsf{crs}_1, \boldsymbol{c}_x, \boldsymbol{a}_0, \boldsymbol{a}_1, \pi_1)$.
   - Output abort if $b = 0$
   - $\boldsymbol{e}' \leftarrow R(\chi_{\sigma'})^{1\times\ell}$.
   - $\boldsymbol{d}_x = \boldsymbol{c}_x \cdot k + \boldsymbol{e}' \bmod q$.
   - $\pi_2 \leftarrow \mathbb{P}_2(k, \boldsymbol{e}', \boldsymbol{e} : \mathsf{crs}_2, \boldsymbol{c}, \boldsymbol{d}_x, \boldsymbol{c}_x, \boldsymbol{a})$.

   and sends $(\boldsymbol{d}_x, \pi_2)$ to a client $\mathbb{C}$ while outputting $\perp$.
3. On receipt of $(\boldsymbol{d}_x, \pi_2)$ a client $\mathbb{C}$ executes
   - $b \leftarrow \mathbb{V}_2(\mathsf{crs}_0, \mathsf{crs}_2, \boldsymbol{c}, \boldsymbol{d}_x, \boldsymbol{c}_x, \pi_2)$.
   - Output abort if $b = 0$.
   - $\boldsymbol{y}_x = \lfloor \boldsymbol{d}_x - \boldsymbol{c} \cdot s \rceil_p$.
   - Output $\boldsymbol{y}_x$.

Figure 2. VOPRF construction

3.1 Zero-Knowledge Argument of Knowledge Statements

The arguments of $\mathbb{P}_i$ algorithms fall into two groups separated by a colon. Arguments before a colon are intended as "secret" information pertaining to a witness for a statement. Arguments after a colon should be interpreted as "public" information specifying the statement that is being proved.

Client Proof. The client proof denoted $\mathbb{P}_1(x, s, \boldsymbol{e}_1 : \mathsf{crs}_1, \boldsymbol{c}_x, \boldsymbol{a}, \boldsymbol{a}_0, \boldsymbol{a}_1)$ should prove knowledge of

- $x \in \{0,1\}^L$
- $s \in R$ where $\|s\|_\infty \leq \sigma \cdot \sqrt{n}$
- $\boldsymbol{e}_1 \in R^{1\times\ell}$ where $\|\boldsymbol{e}_1\|_\infty \leq \sigma\sqrt{n}$

such that $\boldsymbol{c}_x = \boldsymbol{a} \cdot s + \boldsymbol{e}_1 + \boldsymbol{a}_x \bmod q$.

Server Proofs. The server proof in the *initialisation phase* denoted $\mathbb{P}_0(k, \boldsymbol{e} : \mathsf{crs}_0, \boldsymbol{c})$ has the purpose of proving knowledge of $k \in R, \boldsymbol{e} \in R^{1\times\ell}$ where $\|k\|_\infty$, $\|\boldsymbol{e}\|_\infty \leq \sigma \cdot \sqrt{n}$ such that $\boldsymbol{c} = \boldsymbol{a} \cdot k + \boldsymbol{e} \bmod q$ where crs_0 contains $\boldsymbol{a}$.
The server proof in the *query phase* denoted by $\mathbb{P}_2(k, \boldsymbol{e}', \boldsymbol{e} : \mathsf{crs}_2, \boldsymbol{c}, \boldsymbol{d}_x, \boldsymbol{c}_x, \boldsymbol{a})$ has the purpose of proving that there is some

- $k \in R$ where $\|k\|_\infty \leq \sigma \cdot \sqrt{n}$
- $\boldsymbol{e} \in R^{1\times\ell}$ where $\|\boldsymbol{e}\|_\infty \leq \sigma \cdot \sqrt{n}$
- $\boldsymbol{e}' \in R^{1\times\ell}$ where $\|\boldsymbol{e}'\|_\infty \leq \sigma' \cdot \sqrt{n}$

such that

$$\begin{aligned} \boldsymbol{c} &= \boldsymbol{a} \cdot k + \boldsymbol{e} \bmod q, \\ \boldsymbol{d}_x &= \boldsymbol{c}_x \cdot k + \boldsymbol{e}' \bmod q. \end{aligned} \tag{1}$$

It is important to note that both $\boldsymbol{c}$ and $\boldsymbol{d}_x$ each consist of ℓ ring elements. Therefore, the above system consists of a total of 2ℓ noisy products of public ring elements and k. Note that the well-definedness of normal form RLWE (where the secret is drawn from the error distribution) implies that the witnesses used by the prover in π_0 and π_2 share the same value k.

3.2 Optimisations

Removing $\mathbb{P}_0$ using Trapdoors. The main purpose of proof system 0 is to allow the security proof to extract k and forward it on to the functionality. On removing this proof, if the server does not commit to its key properly, it cannot carry out the zero-knowledge proof in the **Query** phase, leading to a protocol where no evaluations are given to clients. An alternative to the server's NIZKAoK in the **Init-S** phase, the proof could extract k via trapdoors. Using the methods of Micciancio and Peikert [46], one can sample a trapdoored $\boldsymbol{a} \in R_q^m$ for $m = \mathcal{O}(\ell)$ that is indistinguishable from uniform where the trapdoor permits

efficient inversion of the function $g_{\boldsymbol{a}}(k, \boldsymbol{e}) = \boldsymbol{a} \cdot k + \boldsymbol{e}$ for small $\boldsymbol{e}$. Therefore, the malicious server security proof could extract k in the **Init-S** phase by using a trapdoored $\boldsymbol{a}$ along with the inversion algorithm. For clarity and simplicity, we do not incorporate these ideas directly into our protocol.

Truncating the PRF. Although the protocol in Figure 2 is concerned with the evaluation of the full BP14 PRF, we may consider a truncated version of the PRF to improve efficiency. In particular, the the BP14 PRF is evaluated as $F_k(x) := \lfloor \boldsymbol{a}_x \cdot k \rceil_p \in R_p^{1\times\ell}$ but we could easily truncate particular quantities in our protocol to consider the PRF $F'_k(x) := \lfloor a_x \cdot k \rceil_p$ where a_x is the ring element appearing in the first entry of $\boldsymbol{a}_x$. The relevant values that are truncated from ℓ ring elements to a single ring element from our protocol are $\boldsymbol{c}, \boldsymbol{a}_x, \boldsymbol{c}_x, \boldsymbol{d}_x, \boldsymbol{y}_x$. Ignoring the zero-knowledge elements of the protocol, this saves us a factor of ℓ. However, computation of the full $\boldsymbol{a}_x$ must still be performed by the client in order to calculate the truncated value. Additionally, the computation of $\boldsymbol{a}_x$ will still need to be considered by the client's zero-knowledge proof. As we will see in Section 4 and Appendix C, the computation of $\boldsymbol{a}_x$ is the main source of inefficiency in the zero-knowledge proofs and our overall protocol. Therefore, we do not trivially save a factor of ℓ in computation time and zero-knowledge proof size by using a truncated BP14 PRF.

Batching Queries. We can save on the cost of zero-knowledge proof of the server in the **Query** phase by batching VOPRF queries. When the client sends a single value $\boldsymbol{c}_x$, the server proves that $\boldsymbol{c}$ and $\boldsymbol{c}_x$ are computed with respect to the same k. If the client sends N individual queries, the server proves that $\boldsymbol{c}$ and $\boldsymbol{c}_{x_1}$ are with respect to the same k and then independently proves that $\boldsymbol{c}$ and $\boldsymbol{c}_{x_2}$ are with respect to the same k and so on. Instead, the server could simply prove that $\boldsymbol{c}, \boldsymbol{c}_{x_1}, \ldots, \boldsymbol{c}_{x_N}$ are all with respect to the same k in one shot, saving an *additive* term of $\mathcal{O}(N \cdot \ell)$ in communication over N different VOPRF evaluations (although the overall complexity of the communication does not change asymptotically).

Cut-and-choose. Another way in which we can improve efficiency (from the server's perspective) is to remove some of the zero-knowledge proofs using a cut-and-choose methodology. In particular, we can remove the need for the zero-knowledge proof from the server in the **Query** phase as follows. Firstly, in the **Init-S** phase, we make the server publish (for small k) the value $\boldsymbol{y} := \lfloor \boldsymbol{a}_{x'} \cdot k \rceil_p$ for some fixed x' in addition to the value $\boldsymbol{a} \cdot k + \boldsymbol{e}$ as well as a zero-knowledge proof attesting to the correct computation of these values for small k. The next change comes in the client message in the **Query** phase. Instead of sending a single pair $(\boldsymbol{c}_x, \pi_1)$, the client chooses a uniform subset X of $\{1, \ldots, N\}$ of size K. The client then sends N pairs $(\boldsymbol{c}_{x_1}, \pi_{1,1}), \ldots, (\boldsymbol{c}_{x_N}, \pi_{1,N})$ where for all $j \in X$, $x_j = x'$ and for all $j' \notin X$, $x_{j'} = x$ for some x chosen by the client. The server then computes $\boldsymbol{d}_{x_1}, \ldots, \boldsymbol{d}_{x_N}$ as it does in Figure 2 using $\boldsymbol{c}_{x_1}, \ldots, \boldsymbol{c}_{x_N}$ respectively.

Next, the client processes each $\boldsymbol{d}_{x_i}$ individually to compute the values $\boldsymbol{y}_{x_1} \dots \boldsymbol{y}_{x_N}$ as in the plain protocol. Finally, the client aborts if any of the following hold:

- there exists a $j^* \in X$ such that $\boldsymbol{y}_{x_{j^*}} \neq \boldsymbol{y}$
- $\boldsymbol{y}_{x_{j'}}$ are not all equal for $j' \notin X$
- $\boldsymbol{y}_{x_{j'}} = \boldsymbol{y}$ for all $j' \notin X$ (see explanation below)

Otherwise, the client accepts $\boldsymbol{y}_x = \boldsymbol{y}_{x_{j'}}$ for any $j' \notin X$ as the evaluation at x. The client now must create N proofs for the most complex statements. On the other hand, the server does not need to create any proofs whatsoever in the online phase. The only way for the server to cheat now is to somehow guess the $N - K$ transcripts containing input x which can be done with probability at most $1/\binom{N}{K}$. Thus, the computational burden is mostly shifted to the client, which might be desirable in some settings.

On close inspection, there is a slight problem with the cut-and-choose optimisation described above. The issue is that a client might ask for an evaluation on input x such that $\lfloor \boldsymbol{a}_x \cdot k \rceil_p = \lfloor \boldsymbol{a}_{x'} \cdot k \rceil_p$ in which case the third condition causes an abort, even though the client obtained the correct evaluation. One way to get around this is to redefine the PRF slightly so that such collisions only occur with negligible probability. For example, for $L-1$ bit inputs $x \in \{0,1\}^{L-1}$, suppose we use the alternative PRF $F'_k(x) := \lfloor \boldsymbol{a}_{0\|x} \cdot k \rceil_p$. Since we can rewrite $\boldsymbol{a}_{0\|x} \cdot k = \boldsymbol{a}_0 \cdot \boldsymbol{Z}_{x,k}$ where $\boldsymbol{Z}_{x,k}$ has small entries as long as k is short. Then a collision in this PRF must lead to an equation $\boldsymbol{a}_0 \cdot (\boldsymbol{Z}_{x,k} - \boldsymbol{Z}_{x',k}) = \boldsymbol{u} \bmod q$ where $\|\boldsymbol{u}\|_\infty \leq q/p$. Rearranging, this equation becomes $[1|\boldsymbol{a}_0] \cdot \begin{bmatrix} \boldsymbol{u} \\ (\boldsymbol{Z}_{x,k} - \boldsymbol{Z}_{x',k}) \end{bmatrix} = \boldsymbol{0} \bmod q$ which means that such a collision would imply a solution to a ring-SIS problem with respect to $[1|\boldsymbol{a}]$ (in Hermite normal form). Therefore, for fixed x and any short k, it is unlikely that a collision in this alternative PRF will occur under some SIS assumption.

3.3 Correctness

Before proving correctness, we present a lemma that we will apply below. The proof of this lemma is in Appendix B.2.

Lemma 5. *Fix any $x \in \{0,1\}^L$. Suppose there exists a PPT algorithm $\mathcal{D}(x, \boldsymbol{a}_0, \boldsymbol{a}_1)$ that outputs $r \in R$ such that $\|r\|_\infty \leq B$ and at least one coefficient of $\boldsymbol{a}_x \cdot r$ is in the set $(q/p) \cdot \mathbb{Z} + [-T, T]$ with non-negligible probability (over a uniform choice of $\boldsymbol{a}_0, \boldsymbol{a}_1 \leftarrow R_q^\ell$ and its random coins). Then there exists an efficient algorithm solving $\mathsf{1D\text{-}SIS}_{q/p,n\ell,\max\{n\ell B,T\}}$ with non-negligible probability.*

Lemma 6 (Correctness). *Adopt the notation of Figure 2, assuming an honest client and server. Define $T := 2\,\sigma^2\, n^2 + \sigma'\sqrt{n}$. For any $x \in \{0,1\}^L, k \in R_q$ such that $\|k\|_\infty \leq \sigma \cdot \sqrt{n}$, we have that*

$$\Pr[\boldsymbol{y}_x \neq F_k(x)] \leq \mathsf{negl}(\kappa)$$

over the choice of PRF parameters $\boldsymbol{a}_0, \boldsymbol{a}_1 \leftarrow R_q^{1\times\ell}$ assuming the hardness of $\mathsf{1D\text{-}SIS}_{q/p,n\ell,T}$.

Proof. Fix an arbitrary x. Assume that there exists a k' such that $\|k'\| \leq \sigma \cdot \sqrt{n}$ where $\Pr[\boldsymbol{y}_x \neq F_{k'}(x)]$ is non-negligible over the choice of $\boldsymbol{a}_0, \boldsymbol{a}_1 \leftarrow R_q^{1\times\ell}$. Expanding $\boldsymbol{c}$ and $\boldsymbol{d}_x$ from the protocol, we have that

$$\boldsymbol{y}_x = \lfloor \boldsymbol{a}_x \cdot k' + \boldsymbol{e}_1 \cdot k' + \boldsymbol{e}' - \boldsymbol{e} \cdot s \rceil_p .$$

Note that $\boldsymbol{e}'' := \boldsymbol{e}_1 \cdot k' - \boldsymbol{e} \cdot s + \boldsymbol{e}'$ has infinity norm less than T as defined in the lemma statement with all but negligible probability. It follows that there must be that at least one coefficient of $\boldsymbol{a}_x \cdot k'$ in the set $(q/p) \cdot \mathbb{Z} + [T, T]$ with non-negligible probability, otherwise $\boldsymbol{y}_x = \lfloor \boldsymbol{a}_x \cdot k' \rceil_p =: F_{k'}(x)$. Applying Lemma 5 to the algorithm $\mathcal{D}(x, \boldsymbol{a}_0, \boldsymbol{a}_1)$ that ignores $\boldsymbol{a}_0, \boldsymbol{a}_1$ and simply outputs k' implies an efficient algorithm solving $\mathsf{1D\text{-}SIS}_{q/p, n\ell, \max\{n^{3/2}\ell\sigma, T\}}$. □

The remainder of the security proof can be found in Section 5.

4 Lattice-based NIZKAoK Instantiations

We now describe various instantiations of our zero-knowledge arguments of knowledge. Note that we use the Fiat-Shamir transform (on parallel repetitions) to obtain non-interactive proofs. We recall that the Fiat-Shamir transform has recently been shown to be secure in the QROM [25,42] in certain settings. We place most of our attention on discussing how to instantiate Proof System 1, as the other proof systems may be derived straight-forwardly using a subset of the techniques arising in Proof System 1. For more precise details on how to instantiate Proof System 1 using the protocol of Yang et al. [58], see Appendix C. Alternatively, one could use the same techniques as in [41] to represent the statement of interest in Proof System 1 as a permuted kernel problem and use the recent protocol of Beullens [7]. The advantage of doing so would be that the protocol of Beullens has been *shown* to be compatible with the aforementioned security results of the Fiat-Shamir transform in the QROM.

Note that the argument system of Yang et al. requires the modulus q to be a prime power. In contrast, 1D-SIS is known to be at least as hard as standard lattice problems when q has many large coprime factors [15]. In order to justify the use of a prime power modulus along with the use of the 1D-SIS assumption, we apply Lemmas 12 and 13 from Appendix B. Alternatively, if one wished to use a highly composite modulus, then a Stern-based protocol such as in [40,41] or the more efficient recent protocol of Beullens [7] may still be used. Nonetheless, all of the aforementioned argument systems involve rewriting PRF evaluations as a large system of linear equations. In our context, applying the argument system of Yang is slightly simpler. Additionally, a single execution of the protocol of Yang et al. achieves a soundness error of $2/(2\bar{p}+1)$ for some polynomial $\bar{p}$ much less than q. This is similar to the soundness error encountered in the Beullens protocol, but significantly improves on the soundness of Stern-based protocols. Therefore, roughly $\kappa/\log \bar{p}$ repetitions are required to reach a $2^{-\kappa}$ soundness error when using either of the protocol of Yang et al. or Beullens protocols.

Proof System 0: Small Secret RLWE Sample. Let $A \in \mathbb{Z}_q^{n\ell \times n}$ be the vertical concatenation of the negacyclic matrices associated to multiplication by the ring elements of $\boldsymbol{a} \in R_q^{1\times \ell}$ respectively. Further, let $\vec{c} \in \mathbb{Z}_q^{n\ell}$ be the vertical concatenation of coefficient vectors of ring elements in $\boldsymbol{c} \in R_q^{1\times \ell}$ respectively. The first proof aims to prove in zero knowledge, knowledge of a short solution $\vec{x} := (\vec{x}_1, \vec{x}_2)$, where $\|\vec{x}\|_\infty \leq \sigma \cdot \sqrt{n}$ to the system

$$\vec{c} = A \cdot \vec{x}_1 + \vec{x}_2 \bmod q.$$

This is an inhomogeneous SIS problem, so the zero-knowledge proof may be instantiated using either the protocol of Yang et al. or Beullens. Additionally, for this proof system, we may also use the protocol from [12]. All of these options avoid the so-called soundness gap seen in many lattice-based proof systems (e.g. [43,44]) although the efficient protocol in [44] has been shown to be secure in the QROM when the Fiat-Shamir transform is applied [42]. Therefore, for simplicity and neatness we prefer to consider these systems when writing the security proof for our VOPRF although one may use the more efficient protocol of [44] in practice.

Proof System 1: Proofs of Masked Partial PRF Computation. This proof system aims to prove that for a known $\boldsymbol{a}$ and $\boldsymbol{c}$, the prover knows short s and $\boldsymbol{e}$ along with a bit-string x such that $\boldsymbol{c} = \boldsymbol{a} \cdot s + \boldsymbol{e} + \boldsymbol{a}_x$ where $\boldsymbol{a}_x$ is part of the BP14 PRF evaluation. At a high level, we will run the protocol of Yang et al. [58] $\mathcal{O}(\kappa/\log \bar{p})$ times (for some $\bar{p} = \mathsf{poly}(\kappa)$) in parallel and apply the Fiat-Shamir heuristic. We focus on this instantiation for simplicity. We do not actually concretely present any ZKAoK protocol in this work, but we do highlight the reduction (Appendix C) showing that we may use the protocol of Yang et al. Similar methods (e.g. the decomposition-extension framework used by[41]) can be used to prove compatibility with the protocol of Beullens. Let P_n represent the power set of $\{1, \ldots, n\}$[3]. The protocol of Yang et al. is a ZKAoK for the instance-witness set given by

$$\mathcal{R}^* = \left\{ \left((\boldsymbol{A}, \vec{y}, \mathcal{M}), \vec{x} \right) \in \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m \times P_n \times \mathbb{Z}_q^n : \substack{\boldsymbol{A}\cdot\vec{x} = \vec{y} \bmod q \ \wedge \\ \forall (i,j,k) \in \mathcal{M}, x_i = x_j \cdot x_k \bmod q} \right\}.$$

Therefore, in order to show that we may use the protocol, we simply reduce our statement of interest to an instance $((\boldsymbol{A}', \vec{y}', \mathcal{M}'), \vec{w}') \in \mathcal{R}^*$. Then, the protocol of Yang et al. allows to argue knowledge of a witness $\vec{w}'$ such that $((\boldsymbol{A}', \vec{y}', \mathcal{M}'), \vec{w}') \in \mathcal{R}^*$. Details on reducing statements of the relevant form to instances in $\mathcal{R}^*$ are given in Appendix C, but a high level overview follows.

First note that we can compute $\boldsymbol{a}_x$ recursively (similarly to [41]) by setting variables $B_i \in R_q^{\ell\times\ell}$ for $i = L-1, \ldots, 0$ via $B_{L-1} = G^{-1}(\boldsymbol{a}_{x_{L-1}})$, and $B_i = G^{-1}(\boldsymbol{a}_{x_i} \cdot B_{i+1})$ for $i = L-2, \ldots, 0$. Using this, we have $\boldsymbol{a}_x = G \cdot B_0$. We can therefore use the system $G \cdot B_i = \boldsymbol{a}_{x_i} \cdot B_{i-1}$ to facilitate computation of $\boldsymbol{a}_x$ along with the linear equation $\boldsymbol{c} = \boldsymbol{a}\cdot s + \boldsymbol{e}_1 + G\cdot B_0$ to completely describe the statement being proved. However, the resulting system is over ring elements and is not linear in unknowns. To solve these issues, we simply replace ring multiplication

by integer matrix-vector products and then linearise the resulting system (which places quadratic constraints amongst the entries of the solution). We also make use of binary decompositions to bound the infinity norms of valid solutions and ensure that necessary entries are in $\{0,1\}$ via quadratic constraints. [11]

Proof System 2: Proofs of Secret Equivalence. Recall that we wish to prove existence of a solution to Equations (1). Note that $\boldsymbol{d}_x$ from the protocol in Section 3 are vectors holding ℓ ring elements. Therefore, Equations (1) can be expressed as a system

$$\begin{aligned} c_i &= a_i \cdot k + e_i && i = 1, \ldots, \ell, \\ (d_x)_i &= (c_x)_i \cdot k + e'_i && i = 1, \ldots, \ell, \end{aligned}$$

where $\|e_i\|_\infty, \|k\|_\infty \leq \sigma \cdot \sqrt{n}$, $\|e'_i\|_\infty \leq \sigma' \cdot \sqrt{n}$. We can conceptualise the above as a large linear system $A' \cdot \vec{x} = \vec{c}$ where $\vec{x}$ is the concatenation of coefficient vectors of $k, e_1, \ldots, e_\ell, e'_1, \ldots, e'_\ell$ and $\vec{c}$ is the concatenation of the coefficient vectors of $c_1, \ldots, c_\ell, (d_x)_1, \ldots, (d_x)_\ell$. Using this interpretation, we may instantiate this proof system using the same methods as in Proof System 0.

5 Security Proof

In this section, we show that the protocol in Figure 2 is a VOPRF achieving security against malicious adversaries. In particular, corrupted clients and servers that attempt to subvert the protocol learn/affect only as much as in an ideal world, where they interact via the functionality $\mathcal{F}_{\mathsf{VOPRF}}$.

Theorem 2. (Security) *Assume $p|q$. The protocol in Figure 2 is a secure VOPRF protocol (according to Definition 1) if the following conditions hold:*

- $\forall i \in \{0,1,2\}, (\mathbb{P}_i, \mathbb{V}_i)$ *is a ZKAoK*
- $\mathsf{dRLWE}_{q,n,\sigma}$ *is hard,*
- $\frac{q}{2p} \gg \sigma' \gg \max\{L \cdot \ell \cdot \sigma n^{3/2}, \sigma^2 n^2\}$,
- $\mathsf{1D\text{-}SIS}_{q/(2p),n\cdot\ell,\max\{\ell\cdot\sigma n^{3/2}, 2\sigma^2 n^2 + \sigma'\sqrt{n}\}}$ *is hard.*

Note that correctness of our protocol with respect to honest clients and servers is shown in Section 3.3. Therefore, what remains is to show average malicious client security and malicious server security.

Correctness of non-aborting malicious protocol runs. During the malicious client proof, it will be useful to call on the fact that a non-aborting protocol transcript enables computation of $F_k(x)$ with overwhelming probability:

[11] Using the fact that $x^2 = x \bmod q \iff x \in \{0,1\}$ assuming q is a prime power.

Lemma 7. *Assume that* $\mathsf{dRLWE}_{q,n,\sigma}$ *is hard,* σ *and* n *are* $\mathsf{poly}(\kappa)$*, and* $\frac{q}{2p} \gg \sigma' \gg \max\{L \cdot \ell \cdot \sigma n^{3/2}, \sigma^2 n^2\}$*. For any* $x \in \{0,1\}^L$*, consider a non-aborting run of the protocol in Figure 2 between a (potentially malicious) efficient client* $\mathbb{C}^*$ *and honest server* $\mathbb{S}$*. Further, let* s *be the value that is extractable from the client's proof in the query phase. Then, the value of* $\lfloor \boldsymbol{d}_x - \boldsymbol{c} \cdot s \rceil_p$ *is equal to* $\lfloor \boldsymbol{a}_x \cdot k \rceil_p$ *with all but negligible probability.*

Proof. We use the notation from Figure 2. First note that for a *non-aborting* protocol run, any *efficient* client $\mathbb{C}^*$ must have produced $\boldsymbol{c}_x$ correctly using some $x \in \{0,1\}^L, s, \boldsymbol{e}_1$ where $\|s\|_\infty, \|\boldsymbol{e}_1\|_\infty \leq \sigma \cdot \sqrt{n}$. Suppose that $\boldsymbol{e}_x \leftarrow \mathcal{E}_{\boldsymbol{a}_0,\boldsymbol{a}_1,x,\sigma}$. We now use the fact that if $\sigma' \gg \max\{L \cdot \ell \cdot \sigma n^{3/2}, \sigma^2 n^2\}$, then $\boldsymbol{e}' \leftarrow R(\chi_{\sigma'})^{1\times\ell}$ and $(\boldsymbol{e}_x - \boldsymbol{e}_1 \cdot k - \boldsymbol{e} \cdot s) + \boldsymbol{e}'$ are statistically close which follows from Lemmas 4 and 2. Therefore, replacing $\boldsymbol{e}'$ by $(\boldsymbol{e}_x - \boldsymbol{e}_1 \cdot k - \boldsymbol{e} \cdot s) + \boldsymbol{e}'$ and noting that $\boldsymbol{c}_x$ must be well-formed due to the NIZKAoK, the client output equation in Figure 2 can be written as

$$\left\lfloor \frac{p}{q}(\boldsymbol{d}_x - \boldsymbol{c} \cdot s) \right\rceil = \left\lfloor \frac{p}{q}(\boldsymbol{a}_x \cdot k + \boldsymbol{e}_x) + \frac{p}{q}\boldsymbol{e}' \right\rceil$$

To complete the proof, we will use the fact that $\frac{p}{q}(\boldsymbol{a}_x \cdot k + \boldsymbol{e}_x)$ is computationally indistinguishable from uniform random over $\frac{p}{q}R_q^{1\times\ell}$ when $\boldsymbol{e}_x \leftarrow \mathcal{E}_{a_0,a_1,x,\sigma}$ assuming the hardness of $\mathsf{dRLWE}_{q,n,\sigma}$ (Lemma 3). This implies that every coefficient in $\frac{p}{q}(\boldsymbol{a}_x k + \boldsymbol{e}_x)$ is at least T away from $\mathbb{Z} + 1/2$ with all but negligible probability for any $T \ll 1$. Setting $T = \frac{p}{q}\left(\sigma' \cdot \sqrt{n} + L \cdot \ell \cdot \sigma n^{3/2}\right) \ll 1$ ensures that $T \leq \frac{p}{q} \cdot \|\boldsymbol{e}_x + \boldsymbol{e}'\|_\infty$ with all but negligible probability. It then follows that

$$\left\lfloor \frac{p}{q}(\boldsymbol{a}_x \cdot k + \boldsymbol{e}_x) + \frac{p}{q}\boldsymbol{e}' \right\rceil = \left\lfloor \frac{p}{q}\boldsymbol{a}_x \cdot k \right\rceil$$

as required. □

5.1 Malicious Client Proof

Lemma 8 (Average-case malicious client security). *Assume that* σ *and* n *are* $\mathsf{poly}(\kappa)$*, and* $p|q$*, and let conditions (i) and (ii) be as follows:*

(i) $\mathsf{dRLWE}_{q,n,\sigma}$ *is hard,*
(ii) $\frac{q}{2p} \gg \sigma' \gg \max\{L \cdot \ell \cdot \sigma n^{3/2}, \sigma^2 n^2\}$.

If the above conditions hold and $(\mathbb{P}_1, \mathbb{V}_1)$ *is a NIZKAoK, then the protocol in Figure 2 has average-case security against malicious clients according to Definition 1.*

Proof. We describe a simulation $\mathcal{S}$ that communicates with the functionality $\mathcal{F}_{\mathsf{VOPRF}}$ (environment) on one hand, and the malicious client $\mathbb{C}^*$ on the other. $\mathcal{S}$ carries out the following steps:

1. During **CRS SetUp**, publish honest $\boldsymbol{a}, \boldsymbol{a}_0, \boldsymbol{a}_1, \mathsf{crs}_1$ and (dishonest) simulated versions of crs_0 and crs_2. Denote the simulated CRS elements crs_0' and crs_2'.

2. Pass the init message onto $\mathcal{F}_{\mathsf{VOPRF}}$, then send $\mathbb{C}^*$ a uniform $\boldsymbol{c} \leftarrow R_q^{1\times\ell}$ with a simulated proof $\pi_{0,\mathsf{Sim}}$. Initialise an empty list received.
3. During the **Query** stage, for each message $(\boldsymbol{c}_x, \pi_1)$ from $\mathbb{C}^*$, do:
   (a) $b \leftarrow \mathbb{V}_1(\mathsf{crs}_1, \boldsymbol{c}_x, \boldsymbol{a}, \boldsymbol{a}_0, \boldsymbol{a}_1, \pi_1)$. If $b = 0$ send abort to the functionality and abort the protocol with the malicious client. If $b = 1$ continue.
   (b) Extract the values $x, s, \boldsymbol{e}_1$ from π_1 using the ZKAoK extractor and send (query, x) to the functionality.
   (c)
   – If $\mathcal{F}_{\mathsf{VOPRF}}$ aborts:
   $\mathcal{S}$ aborts.
   – If $\mathcal{F}_{\mathsf{VOPRF}}$ returns $\boldsymbol{y} \in R_p^{1\times\ell}$ **and** $\forall \boldsymbol{y}^*, (x, \boldsymbol{y}^*) \notin$ received:
   (i.e. if this is the first time x is queried) uniformly sample

$$\boldsymbol{y}_q \leftarrow R_q^{1\times\ell} \cap \left(\frac{q}{p}\boldsymbol{y} + R_{\leq\frac{q}{2p}}^{1\times\ell}\right)$$

   and do received.add$(x, \boldsymbol{y}_q)$.
   – If $\mathcal{F}_{\mathsf{VOPRF}}$ returns $\boldsymbol{y} \in R_p^{\ell}$ **and** $\exists \boldsymbol{y}^*$s.t.$(x, \boldsymbol{y}^*) \in$ received:
   (i.e. x was previously queried) Then set $\boldsymbol{y}_q = \boldsymbol{y}^*$.
   (d) Next pick $\bar{\boldsymbol{e}}' \leftarrow \chi_{\sigma'}$ and set

$$\bar{\boldsymbol{d}}_x = \boldsymbol{c} \cdot s + \bar{\boldsymbol{e}}' + \boldsymbol{y}_q \bmod q.$$

   Finally, produce a simulated proof $\pi_{2,\mathsf{Sim}}$ using crs_2' and send $(\bar{\boldsymbol{d}}_x, \pi_{2,\mathsf{Sim}})$ to $\mathbb{C}^*$.

We now argue that $\mathbb{C}^*$ cannot decide whether it is interacting with $\mathcal{S}$ or with a genuine server. Firstly, recognise that $(\mathsf{crs}_0', \mathsf{crs}_2')$ is indistinguishable from honestly created $(\mathsf{crs}_0, \mathsf{crs}_2)$. Secondly, the malicious client cannot distinguish the simulator's uniform $\boldsymbol{c}$ sent during the **Init** phase from the real protocol by the $\mathsf{dRLWE}_{q,n,\sigma}$ assumption (condition (i)). This implies that both the **CRS SetUp** and **Init** phases that $\mathcal{S}$ performs are indistinguishable from the real protocol.

The most challenging step is arguing that the simulator's behaviour in the **Query** phase is indistinguishable from the real protocol from the malicious client's point of view. We will analyse the behaviour of the simulator assuming that no abort is triggered. We begin by arguing that the server message $\boldsymbol{d}_x$ in the real protocol with respect to any triple $(x, s, \boldsymbol{e}_1)$ can be replaced by a related message $\boldsymbol{c} \cdot s + \boldsymbol{a}_x \cdot k + \boldsymbol{e}_x + \boldsymbol{e}'''$ where $\boldsymbol{e}_x \leftarrow \mathcal{E}_{\boldsymbol{a}_0,\boldsymbol{a}_1,x,\sigma}$ and $\boldsymbol{e}''' \leftarrow R(\chi_{\sigma'})^{1\times\ell}$ without detection by the following statistical argument. We have that the server response in the *real* protocol has $\boldsymbol{d}_x$ of the form

$$\boldsymbol{c} \cdot s + \boldsymbol{e}_1 \cdot k + \boldsymbol{a}_x \cdot k + \boldsymbol{e}' \tag{2}$$

where $\boldsymbol{e}' \leftarrow R(\chi_{\sigma'})^{1\times\ell}$. By Lemma 2, the message distribution in Equation (2) is statistically indistinguishable from

$$\boldsymbol{a} \cdot k \cdot s + \boldsymbol{e} \cdot s + \boldsymbol{a}_x \cdot k + \boldsymbol{e}'' = \boldsymbol{c} \cdot s + \boldsymbol{a}_x \cdot k + \boldsymbol{e}'' \tag{3}$$

where $\boldsymbol{e}'' \leftarrow R(\chi_{\sigma'})^{1\times\ell}$ due to the fact that $\sigma' \gg \sigma^2 n^2$. By a similar argument along with Lemma 4, the quantity given in Equation (3) is statistically close in distribution to

$$\boldsymbol{c} \cdot s + \boldsymbol{e}''' + (\boldsymbol{a}_x \cdot k + \boldsymbol{e}_x). \tag{4}$$

where $\boldsymbol{e}_x \leftarrow \mathcal{E}_{\boldsymbol{a}_0,\boldsymbol{a}_1,x,\sigma}$ and $\boldsymbol{e}''' \leftarrow R(\chi_{\sigma'})^{1\times\ell}$. Next, using Lemma 3 and condition (i), we have that the bracketed term in Equation (4) is indistinguishable from random over $R_q^{1\times\ell}$ by the hardness of $\mathsf{dRLWE}_{q,n,\sigma}$ (Lemma 3). In particular, from an efficient $\mathbb{C}^*$'s point of view, $\boldsymbol{d}_x$ cannot be distinguished from

$$\boldsymbol{c} \cdot s + \boldsymbol{e}''' + \boldsymbol{u}_x$$

Note that on repeated queries, the errors sampled from $R(\chi_{\sigma'})^{1\times\ell}$ are fresh. The fact that $\mathcal{S}$ samples $\boldsymbol{y}_q$ as a uniformly chosen element of a uniformly chosen interval implies the indistinguishability part of average-case malicious client security.

Next, we show that if the malicious client does indeed compute the correct value from the messages it receives from the honest server (in the real protocol), then it can do the same with the messages that it receives from the simulator. In Lemma 7, we show that a malicious client which does not cause an abort can compute $\lfloor \boldsymbol{a}_x \cdot k \rceil_p$ from the messages it receives from the honest server with all but negligible probability. We now show that this is also the case with the messages it receives from $\mathcal{S}$. Consider $\boldsymbol{y}_q$ sampled by $\mathcal{S}$ and also the corresponding value $\bar{\boldsymbol{d}}_x$. In addition, define $\boldsymbol{e}_{\lfloor\rceil} := \boldsymbol{y}_q - (q/p) \cdot \boldsymbol{y} \in R^{1\times\ell}_{\leq \frac{q}{2p}}$ so that $\boldsymbol{e}_{\lfloor\rceil}$ follows the uniform distribution over $R^{1\times\ell}_{\leq \frac{q}{2p}}$. We have that

$$\left\lfloor \frac{p}{q} \left(\bar{\boldsymbol{d}}_x - \boldsymbol{c} \cdot s \right) \right\rceil = \left\lfloor \boldsymbol{y} + \frac{p}{q} \left(\boldsymbol{e}_{\lfloor\rceil} + \bar{\boldsymbol{e}}' \right) \right\rceil. \tag{5}$$

We also know that with all but negligible probability, that $\|\bar{\boldsymbol{e}}'\|_\infty \leq \sigma'\sqrt{n}$, and that $\|\boldsymbol{e}_{\lfloor\rceil}\|_\infty$ is less than $q/(2p) - T$ with all but negligible probability as long as $T \ll (q/2p)$. Taking $T = \sigma'\sqrt{n}$, we get that with all but negligible probability,

$$\left\| \frac{p}{q} \left(\boldsymbol{e}_{\lfloor\rceil} + \bar{\boldsymbol{e}}' \right) \right\|_\infty \leq \frac{1}{2},$$

implying that the quantity in Equation (5) rounds correctly to $\boldsymbol{y}$ with all but negligible probability. Therefore, both the real protocol and simulator enable correct evaluation of the PRF. □

5.2 Malicious Server Proof

Lemma 9. *Let conditions (i) and (ii) be as follows:*

(i) $\mathsf{dRLWE}_{q,n,\sigma}$ *is hard,*
(ii) $\mathsf{1D\text{-}SIS}_{q/(2p),n\cdot\ell,\max\{\ell\cdot\sigma n^{3/2},2\sigma^2 n^2+\sigma'\sqrt{n}\}}$ *is hard.*

If the above conditions hold and $(\mathbb{P}_0, \mathbb{V}_0)$ *and* $(\mathbb{P}_2, \mathbb{V}_2)$ *are both NIZKAoKs, then the protocol in Figure 2 is secure in the presence of malicious servers.*

Proof. We construct a simulator $\mathcal{S}$ interacting with the malicious server $\mathbb{S}^*$ on one hand and with the functionality $\mathcal{F}_{\mathsf{VOPRF}}$ on the other. The simulator $\mathcal{S}$ behaves as follows:

1. During the CRS.SetUp phase, publish honest $\boldsymbol{a}, \boldsymbol{a}_0, \boldsymbol{a}_1, \mathsf{crs}_0, \mathsf{crs}_2$ and (dishonest) simulated crs_1' to use with the proof systems.
2. During the **Init-C** phase, if $\mathbb{S}^*$ sends $\boldsymbol{c} \in R_q^{1\times\ell}$ and an accepting proof π_0, then use the zero knowledge extractor to obtain a key k' from π_0 and forward this on to the functionality. If the message is not of the correct format, or the proof does not verify, then abort.
3. During the **Query** phase, select a uniform random value $\boldsymbol{u} \leftarrow R_q^{1\times\ell}$, and using the ZK simulator, produce a simulated proof $\pi_{1,\mathsf{Sim}}$ using crs_1'. Send the message $(\boldsymbol{u}, \pi_{1,\mathsf{Sim}})$. Wait for a response of the form $(\widetilde{\boldsymbol{d}}_x, \widetilde{\pi}_2)$ from $\mathbb{S}^*$. If the proof $\widetilde{\pi}_2$ verifies, forward on $\mathsf{deliver}$ to $\mathcal{F}_{\mathsf{VOPRF}}$. Otherwise, forward abort to $\mathcal{F}_{\mathsf{VOPRF}}$.

We will show that the joint output of an honest client $\mathbb{C}$ and $\mathbb{S}^*$ in the real world (where they interact directly) and the ideal world (where they interact via $\mathcal{F}_{\mathsf{VOPRF}}$ and $\mathcal{S}$) are computationally indistinguishable. We begin by arguing that the malicious server $\mathbb{S}^*$ cannot distinguish whether it is interacting with a real client or $\mathcal{S}$, as described above. Firstly, replacing crs_1 by crs_1' is indistinguishable from the point of view of $\mathbb{S}^*$ by definition of a simulated CRS. Importantly, if $\mathbb{S}^*$ can produce valid proofs in the **Init** phase, the key k' obtained by the simulator is the *unique* ring element consistent with c by the uniqueness of normal form RLWE solutions.

All that is left to consider is the **Query** phase. Note that in the real protocol, the client produces $\boldsymbol{c}_x$ which takes the form of a RLWE sample offset by some independent value. This implies that the value $\boldsymbol{c}_x$ is pseudorandom under the hardness of $\mathsf{dRLWE}_{q,n,\sigma}$. Therefore, the malicious server $\mathbb{S}^*$ cannot distinguish a real $\boldsymbol{c}_x$ from the pair $\boldsymbol{u}$ that $\mathcal{S}$ uses. By the properties of a ZK simulator, it follows that a real client message $(\boldsymbol{c}_x, \pi_1)$ and crs_1 is indistinguishable from $(\boldsymbol{u}, \pi_{1,\mathsf{Sim}})$ and crs_1'. Next, if the response from $\mathbb{S}^*$ has a valid proof, then $\mathcal{S}$ forwards on $\mathsf{deliver}$. This means that the ideal functionality passes a PRF evaluation to the client using the server key k'. We now argue that this emulates the output on the client side when running the real protocol with malicious server $\mathbb{S}^*$.

The case where the proof verification fails is trivial since the client aborts in the real and ideal worlds. As a result, we focus on the case where the zero knowledge proof produced by $\mathbb{S}^*$ verifies correctly. Let $s \leftarrow R(\chi_\sigma)$ and $\boldsymbol{e}_1 \leftarrow R(\chi_\sigma)^{1\times\ell}$ be sampled by the honest client. For this honest client interacting with malicious $\mathbb{S}^*$ in the real protocol, observe that

$$\frac{p}{q}\left(\boldsymbol{d}_x - \boldsymbol{c}\cdot s\right) = \frac{p}{q}\boldsymbol{a}_x \cdot k' + \frac{p}{q}(\boldsymbol{e}_1 \cdot k' - \boldsymbol{e}\cdot s + \boldsymbol{e}') \tag{6}$$

for $k', \boldsymbol{e}'$ chosen by $\mathbb{S}^*$ where $\|k'\|_\infty \leq \sigma\cdot\sqrt{n}$ and $\|\boldsymbol{e}'\|_\infty \leq \sigma'\cdot\sqrt{n}$. Therefore, rounding the quantity in Equation (6) is guaranteed to result in the correct value

if every coefficient of $\frac{p}{q} \cdot \boldsymbol{a}_x \cdot k'$ is further than

$$\left\| \frac{p}{q}(\boldsymbol{e}_1 \cdot k' - \boldsymbol{e} \cdot s + \boldsymbol{e}') \right\|_\infty$$

away from $\mathbb{Z} + 1/2$. In other words if $\mathbb{S}^*$ can force incorrect evaluation, it has found $k' \leq \sigma \cdot \sqrt{n}$ such that a coefficient of $\boldsymbol{a}_x \cdot k'$ is within a distance

$$\left\| \boldsymbol{e}_1 \cdot k' - \boldsymbol{e} \cdot s + \boldsymbol{e}' \right\|_\infty \leq 2\sigma^2 n^2 + \sigma' \sqrt{n}$$

of $\frac{q}{p}\mathbb{Z} + \frac{q}{2p} \subset \frac{q}{2p}\mathbb{Z}$. We now apply Lemma 5 with $2 \cdot p$, $T = 2\sigma^2 n^2 + \sigma' \sqrt{n}$ to show that $\mathbb{S}^*$ forcing incorrect evaluation with non-negligible probability violates the assumption that $\mathsf{1D\text{-}SIS}_{q/2p, n \cdot \ell, \max\{\ell \cdot \sigma n^{3/2}, 2\sigma^2 n^2 + \sigma' \sqrt{n}\}}$ is hard. Therefore, condition (ii) enforces correct evaluation. □

5.3 Setting Parameters

Let κ be the security parameter. Ignoring the ZKAoK requirements for simplicity, Theorem 2 requires the following conditions:

- $\mathsf{dRLWE}_{q,n,\sigma}$ is hard,
- $\frac{q}{2p} \gg \sigma' \gg \max\{L \cdot \ell \cdot \sigma n^{3/2}, \sigma^2 n^2\}$,
- $\mathsf{1D\text{-}SIS}_{q/(2p), n \cdot \ell, \max\{\ell \cdot \sigma n^{3/2}, 2\sigma^2\, n^2 + \sigma' \sqrt{n}\}}$ is hard.

We will be using the presumed hardness of SIVP_γ for approximation factors $\gamma = 2^{o(\sqrt{n})}$. The SIVP_γ lattice dimension associated to RLWE will be $n = \kappa^c$ (for some constant $c > 2$); the dimension associated to 1D-SIS hardness will be $n' = \kappa$. We first choose $L = \kappa, \sigma = \mathsf{poly}(n)$ and $\sigma' = \sigma^2 n^2 \cdot \kappa^{\omega(1)}$, and then set $q = p \cdot \prod_{i=1}^{n'} p_i$ by picking coprime $p, p_1, \ldots, p_{n'} = \sigma' \cdot \omega(\sqrt{n n' \log q \log n'})$. Having made these choices, we argue that each of the three conditions are satisfied. We can apply Theorem 3 to argue RLWE hardness via SIVP for sub-exponential approximation factors $2^{\widetilde{\mathcal{O}}(n^{1/c})}$ (for $c > 2$), noting that $\sigma = \mathsf{poly}(n)$ and

$$\begin{aligned} q &= (\sigma')^{n'} \cdot \omega((n \cdot n' \cdot \log q \cdot \log n')^{n'/2}) \\ &= 2^{(2\log(n\sigma) + \omega(1)\log\kappa) \cdot n^{1/c}} \cdot \omega((n \cdot n' \cdot \log q \cdot \log n')^{n'/2}) \\ &= 2^{\omega(1) \cdot n^{1/c} \cdot \log n} \cdot \omega((n^{1+\frac{1}{c}} \cdot \log q \cdot \log n)^{n^{1/c}/2}) \\ &= 2^{\widetilde{\mathcal{O}}(n^{1/c})}. \end{aligned}$$

Now substituting in $\ell = \log q$ implies that the second condition can be satisfied. Finally for the 1D-SIS condition, we note that $q/p = \prod_{i=1}^{n'} p_i$ and

$$\begin{aligned} p_1 &= \sigma' \cdot \omega(\sqrt{n \cdot n' \log q \cdot \log n'}) \\ &= \sigma^2 n^2 \cdot \kappa^{\omega(1)} \cdot \omega(\sqrt{n \cdot n' \cdot \log q \cdot \log n'}) \\ &= (n')^{\omega(1)} \cdot \omega(\sqrt{n'^{1+c} \cdot \log q \cdot \log n'}). \end{aligned}$$

Table 1. Parameters of our VOPRF

Parameter	Description	Requirement	Asymptotic
n	ring dimension	$n = \mathsf{poly}(\kappa)$	$\mathsf{poly}(\kappa)$
q	original modulus	$q = p \cdot \sigma' \cdot \kappa^{\omega(1)}$	$\kappa^{\omega(1)}$
p	rounding modulus	—	$\mathsf{poly}(\kappa)$
ℓ	$\log_2(q)$	—	$\omega(1)$
σ	secret/error distribution	$q/\sigma = 2^{o(\sqrt{n})}$	$\mathsf{poly}(\kappa)$
σ'	drowning distribution	$\sigma' = \sigma^2 n^2 \cdot \kappa^{\omega(1)}$	$\kappa^{\omega(1)}$
L	bit-length of PRF input	—	—

So applying Lemma 10, we get hardness of our 1D-SIS instance via the presumed hardness of SIVP on n'-dimensional lattices for $(n')^{\omega(1)} \cdot \mathsf{poly}(n')$ approximation factors. We summarise the parameters of our construction in Table 1.

To give a rough estimate for concrete bandwidth costs, we start by observing that we need q to be super-polynomial in κ for (a) PRF correctness and (b) noise drowning on the server side. We may pick $\log q \approx 256$ for $\kappa = 128$. Applying the "estimator" from [1] with the quantum cost model from [2] and noise standard deviation $\sigma = 3.2$ suggests that $n = 16,384$ provides security of $> 2^{128}$ operations (indeed, significantly more, suggesting room for fine tuning). Thus, a single RLWE sample takes about 0.5MB. As specified in Section 3 our construction sends $2\,\ell$ such samples. However, an implementation could send only two such samples (see Secction 3.2). Thus, each party would send about 1MB of RLWE sample material. Of course, a more careful analysis and optimisation – picking parameters, analysing bounds, applying rounding, perhaps removing the need for super-polynomial drowning – would reduce this magnitude.

In addition to this, each party must send material for the zero-knowledge proofs. In Appendix C, we show that the statement associated to the client proof may be written as an instance of $\mathcal{R}^*$ consisting of *more than* $m' = n\ell^2(L-1)$ equations where the witness has a dimension of *more than* $n' = 4n\ell^2(L-1)$. Additionally, there are at least $|\mathcal{M}| := 4n\ell^2(L-1)$ constraints. This implies that the argument system of [58] requires the communication of at least $m' + 3n' + 4|\mathcal{M}| = 9n\ell^2(L-1)$ integers modulo q per repetition. Using the concrete parameters laid out above, we require $> 9 \cdot 16,384 \cdot 256^2 \cdot 127 > 2^{40}$ bits of communication per repetition. We remind the reader that choosing parameters of the ZKAoK of Yang appropriately would allow us to only repeat a small number of times and stress that this discussion gives a crude lower bound designed to give an intuition on the inefficiency of our scheme, rather than a formal analysis of the concrete cost of our scheme. We note that applying a SNARK or STARK would reduce the bandwidth requirement for proofs.

Acknowledgements

We thank Ward Beullens for helpful discussions and advice on replacing Stern-based ZKAoKs with more efficient alternatives.

The research of Albrecht was supported by EPSRC grants EP/S020330/1 and EP/S02087X/1, and by the European Union Horizon 2020 Research and Innovation Program Grant 780701; the research of Deo was partially supported by the EPSRC and the UK government as part of the Centre for Doctoral Training in Cyber Security at Royal Holloway, University of London (EP/K035584/1), the European Union Horizon 2020 Research and Innovation Program Grant 780701, and BPI-France in the context of the national project RISQ (P141580); the research of Smart was supported by ERC Advanced Grant ERC-2015-AdG-IMPaCT and by the FWO under an Odysseus project GOH9718N.

References

1. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. J. Mathematical Cryptology 9(3), 169–203 (2015), http://www.degruyter.com/view/j/jmc.2015.9.issue-3/jmc-2015-0016/jmc-2015-0016.xml
2. Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-quantum key exchange - A new hope. In: Holz, T., Savage, S. (eds.) USENIX Security 2016. pp. 327–343. USENIX Association (Aug 2016)
3. Babai, L.: On Lovász' lattice reduction and the nearest lattice point problem. Combinatorica 6(1), 1–13 (1986), https://doi.org/10.1007/BF02579403
4. Banaszczyk, W.: New bounds in some transference theorems in the geometry of numbers. Mathematische Annalen 296(1) (Dec 1993)
5. Banerjee, A., Peikert, C.: New and improved key-homomorphic pseudorandom functions. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 353–370. Springer, Heidelberg (Aug 2014)
6. Banerjee, A., Peikert, C., Rosen, A.: Pseudorandom functions and lattices. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 719–737. Springer, Heidelberg (Apr 2012)
7. Beullens, W.: Sigma protocols for MQ, PKP and SIS, and Fishy signature schemes. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part III. LNCS, vol. 12107, pp. 183–211. Springer, Heidelberg (May 2020)
8. Biasse, J.F., Espitau, T., Fouque, P.A., Gélin, A., Kirchner, P.: Computing generator in cyclotomic integer rings - A subfield algorithm for the principal ideal problem in $L_{|\Delta_{\mathbb{K}}|}(\frac{1}{2})$ and application to the cryptanalysis of a FHE scheme. In: Coron, J.S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part I. LNCS, vol. 10210, pp. 60–88. Springer, Heidelberg (Apr / May 2017)
9. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications (extended abstract). In: 20th ACM STOC. pp. 103–112. ACM Press (May 1988)
10. Boneh, D., Dagdelen, Ö., Fischlin, M., Lehmann, A., Schaffner, C., Zhandry, M.: Random oracles in a quantum world. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 41–69. Springer, Heidelberg (Dec 2011)
11. Boneh, D., Lewi, K., Montgomery, H.W., Raghunathan, A.: Key homomorphic PRFs and their applications. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 410–428. Springer, Heidelberg (Aug 2013)
12. Bootle, J., Lyubashevsky, V., Seiler, G.: Algebraic techniques for short(er) exact lattice-based zero-knowledge proofs. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part I. LNCS, vol. 11692, pp. 176–202. Springer, Heidelberg (Aug 2019)

13. Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: Classical hardness of learning with errors. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC. pp. 575–584. ACM Press (Jun 2013)
14. Brakerski, Z., Tsabary, R., Vaikuntanathan, V., Wee, H.: Private constrained PRFs (and more) from LWE. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 264–302. Springer, Heidelberg (Nov 2017)
15. Brakerski, Z., Vaikuntanathan, V.: Constrained key-homomorphic PRFs from standard lattice assumptions - or: How to secretly embed a circuit in your PRF. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 1–30. Springer, Heidelberg (Mar 2015)
16. Canetti, R., Chen, Y.: Constraint-hiding constrained PRFs for NC^1 from LWE. In: Coron, J.S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part I. LNCS, vol. 10210, pp. 446–476. Springer, Heidelberg (Apr / May 2017)
17. CFRG: Cfrg pake selection process. Public GitHub repository (Summer 2019), https://github.com/cfrg/pake-selection. Accessed Jan 2020
18. Cramer, R., Ducas, L., Peikert, C., Regev, O.: Recovering short generators of principal ideals in cyclotomic rings. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 559–585. Springer, Heidelberg (May 2016)
19. Cramer, R., Ducas, L., Wesolowski, B.: Short stickelberger class relations and application to ideal-SVP. In: Coron, J.S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part I. LNCS, vol. 10210, pp. 324–348. Springer, Heidelberg (Apr / May 2017)
20. Damgård, I., Pastro, V., Smart, N.P., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 643–662. Springer, Heidelberg (Aug 2012)
21. Davidson, A., Goldberg, I., Sullivan, N., Tankersley, G., Valsorda, F.: Privacy pass: Bypassing internet challenges anonymously. PoPETs 2018(3), 164–180 (2018)
22. Davidson, A., Sullivan, N.: The privacy pass protocol. Internet-Draft draft-privacy-pass-0, IETF Secretariat (November 2019), https://datatracker.ietf.org/doc/draft-privacy-pass/
23. Davidson, A., Sullivan, N., Wood, C.: Oblivious pseudorandom functions (OPRFs) using prime-order groups. Internet-Draft draft-irtf-cfrg-voprf-01, IETF Secretariat (July 2019), http://www.ietf.org/internet-drafts/draft-irtf-cfrg-voprf-01.txt
24. Dodis, Y., Goldwasser, S., Kalai, Y.T., Peikert, C., Vaikuntanathan, V.: Public-key encryption schemes with auxiliary inputs. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 361–381. Springer, Heidelberg (Feb 2010)
25. Don, J., Fehr, S., Majenz, C., Schaffner, C.: Security of the Fiat-Shamir transformation in the quantum random-oracle model. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part II. LNCS, vol. 11693, pp. 356–383. Springer, Heidelberg (Aug 2019)
26. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (Aug 1987)
27. Freedman, M.J., Ishai, Y., Pinkas, B., Reingold, O.: Keyword search and oblivious pseudorandom functions. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 303–324. Springer, Heidelberg (Feb 2005)
28. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC. pp. 197–206. ACM Press (May 2008)

29. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 75–92. Springer, Heidelberg (Aug 2013)
30. Goldreich, O., Micali, S., Wigderson, A.: How to prove all NP-statements in zero-knowledge, and a methodology of cryptographic protocol design. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 171–185. Springer, Heidelberg (Aug 1987)
31. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems (extended abstract). In: 17th ACM STOC. pp. 291–304. ACM Press (May 1985)
32. Hoffstein, J., Howgrave-Graham, N., Pipher, J., Silverman, J.H., Whyte, W.: NTRUSIGN: Digital signatures using the NTRU lattice. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 122–140. Springer, Heidelberg (Apr 2003)
33. Jarecki, S., Kiayias, A., Krawczyk, H.: Round-optimal password-protected secret sharing and T-PAKE in the password-only model. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part II. LNCS, vol. 8874, pp. 233–253. Springer, Heidelberg (Dec 2014)
34. Jarecki, S., Kiayias, A., Krawczyk, H., Xu, J.: Highly-efficient and composable password-protected secret sharing (or: How to protect your bitcoin wallet online). In: EuroS&P. pp. 276–291. IEEE (2016)
35. Jarecki, S., Krawczyk, H., Xu, J.: OPAQUE: An asymmetric PAKE protocol secure against pre-computation attacks. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part III. LNCS, vol. 10822, pp. 456–486. Springer, Heidelberg (Apr / May 2018)
36. Jarecki, S., Liu, X.: Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 577–594. Springer, Heidelberg (Mar 2009)
37. Katz, J., Ostrovsky, R.: Round-optimal secure two-party computation. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 335–354. Springer, Heidelberg (Aug 2004)
38. Keelveedhi, S., Bellare, M., Ristenpart, T.: Dupless: Server-aided encryption for deduplicated storage. In: Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13). pp. 179–194. USENIX, Washington, D.C. (2013)
39. Krawczyk, H.: The opaque asymmetric pake protocol. Internet-Draft draft-krawczyk-cfrg-opaque-02, IETF Secretariat (July 2019), http://www.ietf.org/internet-drafts/draft-krawczyk-cfrg-opaque-02.txt, http://www.ietf.org/internet-drafts/draft-krawczyk-cfrg-opaque-02.txt
40. Libert, B., Ling, S., Mouhartem, F., Nguyen, K., Wang, H.: Zero-knowledge arguments for matrix-vector relations and lattice-based group encryption. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part II. LNCS, vol. 10032, pp. 101–131. Springer, Heidelberg (Dec 2016)
41. Libert, B., Ling, S., Nguyen, K., Wang, H.: Zero-knowledge arguments for lattice-based PRFs and applications to E-cash. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part III. LNCS, vol. 10626, pp. 304–335. Springer, Heidelberg (Dec 2017)
42. Liu, Q., Zhandry, M.: Revisiting post-quantum Fiat-Shamir. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part II. LNCS, vol. 11693, pp. 326–355. Springer, Heidelberg (Aug 2019)
43. Lyubashevsky, V.: Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 598–616. Springer, Heidelberg (Dec 2009)

44. Lyubashevsky, V.: Lattice signatures without trapdoors. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 738–755. Springer, Heidelberg (Apr 2012)
45. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer, Heidelberg (May / Jun 2010)
46. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 700–718. Springer, Heidelberg (Apr 2012)
47. Papadopoulos, D., Wessels, D., Huque, S., Naor, M., Včelák, J., Reyzin, L., Goldberg, S.: Making NSEC5 practical for DNSSEC. Cryptology ePrint Archive, Report 2017/099 (2017), http://eprint.iacr.org/2017/099
48. Peikert, C.: A decade of lattice cryptography. Cryptology ePrint Archive, Report 2015/939 (2015), http://eprint.iacr.org/2015/939
49. Peikert, C., Regev, O., Stephens-Davidowitz, N.: Pseudorandomness of ring-LWE for any ring and modulus. In: Hatami, H., McKenzie, P., King, V. (eds.) 49th ACM STOC. pp. 461–473. ACM Press (Jun 2017)
50. Peikert, C., Shiehian, S.: Privately constraining and programming PRFs, the LWE way. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part II. LNCS, vol. 10770, pp. 675–701. Springer, Heidelberg (Mar 2018)
51. Pellet-Mary, A., Hanrot, G., Stehlé, D.: Approx-SVP in ideal lattices with preprocessing. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part II. LNCS, vol. 11477, pp. 685–716. Springer, Heidelberg (May 2019)
52. Pornin, T., Prest, T.: More efficient algorithms for the NTRU key generation using the field norm. In: Lin, D., Sako, K. (eds.) PKC 2019, Part II. LNCS, vol. 11443, pp. 504–533. Springer, Heidelberg (Apr 2019)
53. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) 37th ACM STOC. pp. 84–93. ACM Press (May 2005)
54. Rotaru, D., Smart, N.P., Tanguy, T., Vercauteren, F., Wood, T.: Actively secure setup for SPDZ. Cryptology ePrint Archive, Report 2019/1300 (2019), https://eprint.iacr.org/2019/1300
55. Stehlé, D., Steinfeld, R.: Making NTRUEncrypt and NTRUSign as secure as standard worst-case problems over ideal lattices. Cryptology ePrint Archive, Report 2013/004 (2013), http://eprint.iacr.org/2013/004
56. Stein, W., et al.: Sage Mathematics Software Version 9.0. The Sage Development Team (2019), available at http://www.sagemath.org
57. Sullivan, N.: Cloudflare supports privacy pass. Cloudflare Blog (November 09 2017), https://blog.cloudflare.com/cloudflare-supports-privacy-pass/. Accessed Aug 2019
58. Yang, R., Au, M.H., Zhang, Z., Xu, Q., Yu, Z., Whyte, W.: Efficient lattice-based zero-knowledge arguments with standard soundness: Construction and applications. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part I. LNCS, vol. 11692, pp. 147–175. Springer, Heidelberg (Aug 2019)

A Computational Lattice Problems

An n-dimensional lattice Λ is a discrete subgroup of $\mathbb{R}^n$. The i^{th} successive minimum of a lattice Λ, denoted by $\lambda_i(\Lambda)$, is the radius of the smallest ball

centred at the origin containing at least i linearly independent lattice vectors. In addition to the 1D-SIS and RLWE problems, we define the 1D-SISR problem:

Definition 8. ([15, Definition 3.6]) *Let $q = p \cdot \prod_{i\in[n]} p_i$ where $p_1 < \cdots < p_n$ and p are all co-prime. Further, let $m \in \mathbb{N}$. The $\mathsf{1D\text{-}SIS\text{-}R}_{q,p,m,t}$ problem is the following: Given $\boldsymbol{v} \leftarrow \mathbb{Z}_q^m$, find $\boldsymbol{z} \in \mathbb{Z}^m$ with $||\boldsymbol{z}||_\infty \leq t$ such that $\langle \boldsymbol{v}, \boldsymbol{z}\rangle \in [-t, t] + (q/p)\mathbb{Z}$.*

Next we recall some standard lattice problems.

Definition 9. (SVP_γ) *The γ-approximate* shortest vector problem*, denoted SVP_γ, asks that given any basis B of an n-dimensional lattice Λ, and $\gamma = \gamma(n) \geq 1$ that one finds a $\boldsymbol{v} \in \Lambda\backslash\{\mathbf{0}\}$ such that $\|\boldsymbol{v}\|_\infty \leq \gamma \cdot \lambda_1(\Lambda)$.*

Definition 10. (GapSVP_γ) *The γ-gap* shortest vector problem*, denoted GapSVP_γ is the following: Given any basis B of an n-dimensional lattice Λ, $\gamma = \gamma(n) \geq 1$, and $r \in \mathbb{R}_+$, output 1 if $\lambda_1(\Lambda) \leq r$, and 0 if $\gamma \cdot r \leq \lambda_1(\Lambda)$. If $\gamma \cdot r \leq \lambda_1(\Lambda) \leq r$, then any output is acceptable.*

Definition 11. (SIVP_γ) *The γ-*shortest independent vectors problem*, denoted SIVP_γ is the following: Given any basis B of an n-dimensional lattice Λ, find n linearly independent vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$ such that $\max(\|\boldsymbol{v}_i\|_2) \leq \gamma \cdot \lambda_n(\Lambda)$.*

Writing $A \geq B$ to denote that there is a polynomial time reduction from B to A, we rely on the reductions $\mathsf{1D\text{-}SIS\text{-}R}_{q,p,m,t} \geq \mathsf{1D\text{-}SIS}_{q,m,t} \geq (\mathsf{GapSVP}_\gamma, \mathsf{SIVP}_\gamma)$ and $\mathsf{dRLWE}_{q,n,m,\sigma} \geq \mathsf{SIVP}_\gamma$ formalised in the following lemma statements.

Lemma 10. ([15, Corollary 3.5]) *Let $n \in \mathbb{N}$ and $q = \prod_{i\in[n]} p_i$ where all $p_1 < \ldots, < p_n$ are co-prime. Let $m \geq cn \log q$ (for some universal constant c). Assuming that $p_1 \geq t \cdot \omega(\sqrt{mn \log n})$, $\mathsf{1D\text{-}SIS}_{q,m,t}$ is at least as hard as $\mathsf{SIVP}_{t\cdot\widetilde{\mathcal{O}}(\sqrt{mn})}$ and $\mathsf{GapSVP}_{t\cdot\widetilde{\mathcal{O}}(\sqrt{mn})}$.*

Lemma 11. ([15, Corollary 3.7]) *Let q, p, t, m be as in Definition 8. Then the $\mathsf{1D\text{-}SIS\text{-}R}_{q,p,m,t}$ problem is at least as hard as $\mathsf{1D\text{-}SIS}_{q/p,m,t}$. Further, if $p_1 \geq t \cdot \omega(\sqrt{mn \log n})$, then $\mathsf{1D\text{-}SIS\text{-}R}_{q,p,m,t}$ is at least as hard as $\mathsf{SIVP}_{t\cdot\widetilde{\mathcal{O}}(\sqrt{mn})}$ and $\mathsf{GapSVP}_{t\cdot\widetilde{\mathcal{O}}(\sqrt{mn})}$.*

For hardness, we require that the approximation factors $t \cdot \widetilde{\mathcal{O}}(\sqrt{mn})$ be sub-exponential (in the lattice dimension) for the general lattice problems in the corollary above (see below the next lemma). We also recall a reduction from lattice problems to the RLWE problem:

Theorem 3. ([49], Corollary 6.3) *Let $q = q(n) \geq 2$ and $\sigma < q$ be such that $\sigma \geq \omega(1)$. Then $\mathsf{RLWE}_{q,n,\sigma}$ is at least as hard as SIVP_γ over ideal lattices where $\gamma \leq \max\{\omega(\sqrt{n \log n} \cdot q/\sigma), 2\sqrt{n}\}$.*

Previous work [8,18,19,51] shows that the best known algorithms solving SVP_γ for $\gamma = 2^{o(\sqrt{n})}$ have a superpolynomial cost in both the classical and quantum computing models. Therefore, we make the *assumption* that SIVP_γ for $\gamma = 2^{o(\sqrt{n})}$ cannot be solved efficiently.

B Various Results

B.1 Removing the restriction on q for $\mathsf{1D\text{-}SIS}_{q,(\cdot),(\cdot)}$

Note that Lemma 10 only holds for values of q that have many large coprime factors. This can severely limit parametrisation of schemes relying on the hardness of 1D-SIS. Therefore, the following lemma loosens the restriction on q when using the 1D-SIS assumption at the expense of a larger norm bound on acceptable solutions.[12]

Lemma 12. *Let q and q' be integers where $q > q'$ such that the rounding function $\lfloor\cdot\rceil_{q'} : \mathbb{Z}_q \to \mathbb{Z}_{q'}$ maps the uniform distribution over $\mathbb{Z}_q$ to a distribution that is indistinguishable from the uniform distribution over $\mathbb{Z}_{q'}$. Then there is a polynomial time reduction from $\mathsf{1D\text{-}SIS}_{q,m,t}$ to $\mathsf{1D\text{-}SIS}_{q',m,t'}$ where $t = \frac{q}{q'}\left(\frac{m}{2}+1\right)\cdot t'$.*

Proof. Let $\boldsymbol{v} \in \mathbb{Z}_q^m$ be a uniform $\mathsf{1D\text{-}SIS}_{q,m,t}$ instance. By assumption, $\boldsymbol{v}' := \lfloor \boldsymbol{v} \rceil_{q'} \in \mathbb{Z}_{q'}^m$ is statistically close to uniform. In addition, if an algorithm solves the $\mathsf{1D\text{-}SIS}_{q',m,t'}$ problem on instance $\boldsymbol{v}'$, it finds $\boldsymbol{z}'$ such that $\langle \boldsymbol{v}', \boldsymbol{z}' \rangle \in [-t', t'] + q'\mathbb{Z}$ and $\|\boldsymbol{z}'\|_\infty \leq t'$. Multiplying by $\frac{q}{q'}$, we find that $\langle \frac{q}{q'}\boldsymbol{v}', \boldsymbol{z}' \rangle \in [-\frac{q}{q'}t', \frac{q}{q'}t'] + q\mathbb{Z}$. Note that $\boldsymbol{v} = \frac{q}{q'}\boldsymbol{v}' + \boldsymbol{e}$ for some $\|\boldsymbol{e}\|_\infty \leq \frac{q}{2q'}$, which means that $\langle \boldsymbol{e}, \boldsymbol{z}' \rangle \in [-\frac{q}{2q'}mt', \frac{q}{2q'}mt']$. Therefore, $\langle \boldsymbol{v}, \boldsymbol{z}' \rangle = \langle \frac{q}{q'}\boldsymbol{v}', \boldsymbol{z}' \rangle + \langle \boldsymbol{e}, \boldsymbol{z}' \rangle \in [-t, t] + q\mathbb{Z}$. □

We next show that as long as q/q' is negligibly close to an integer, then the output of the rounding function $\lfloor\cdot\rceil_{q'}$ takes a uniform input distribution to an output distribution that is statistically close to uniform. For example, if $q = \kappa^{\omega(1)}$ and $q' = q - \mathsf{poly}(\kappa)$, then we may apply Lemma 12.

Lemma 13. *Let q and p be integers such that $q > p$. Then the distribution $\mathcal{D}$ arising from sampling uniform $v \leftarrow \mathbb{Z}_q$ and outputting $\lfloor v \rceil_p$ is within a statistical distance of $\min\left(1 - \frac{\lfloor q/p \rfloor}{q/p}, \frac{\lceil q/p \rceil}{q/p} - 1\right)$ of the uniform distribution over $\mathbb{Z}_p$.*

Proof. We begin by defining $k = q/p > 1$ and assume that $k \notin \mathbb{Z}$ (otherwise the lemma trivially holds). In doing so, each $w \in \mathbb{Z}_p$ has either $\lceil k \rceil$ or $\lfloor k \rfloor$ rounding pre-images. Let $X \subset \mathbb{Z}_p$ denote the set of elements with $\lceil k \rceil$ pre-images and denote its complement by $\bar{X}$. By the fact that $|X| + |\bar{X}| = p$ and $\lceil k \rceil |X| + \lfloor k \rfloor |\bar{X}| = q$, we can show that $|X|/q = 1 - \lfloor k \rfloor / k$. Denoting $\mathcal{U}$ as the

[12] We are not aware of these lemmas being explicitly stated in the literature, but they seem to be folklore.

uniform distribution over $\mathbb{Z}_p$, we have that

$$\begin{aligned}
2 \cdot \Delta(\mathcal{D}, \mathcal{U}) &= \sum_{x \in X} \left(\frac{\lceil k \rceil}{q} - \frac{k}{q} \right) + \sum_{x \notin X} \left(\frac{k}{q} - \frac{\lfloor k \rfloor}{q} \right) \\
&= 2 \cdot \sum_{x \in X} \left(\frac{\lceil k \rceil}{q} - \frac{k}{q} \right) \\
&= \frac{2|X|}{q} \left(\lceil k \rceil - k \right) \\
&= \frac{2}{k} \cdot (k - \lfloor k \rfloor) \cdot (\lceil k \rceil - k) \\
&< 2 \cdot \min \left(1 - \frac{\lfloor k \rfloor}{k}, \frac{\lceil k \rceil}{k} - 1 \right).
\end{aligned}$$

□

B.2 Proof of Lemma 5

Proof. We will explicate a reduction from the related 1D-SIS-R$_{,,,}$ problem and then use Lemma 11 to complete the proof. Consider the following algorithm $\mathcal{A}$ using $\mathcal{D}$ as a sub-routine that attempts to solve 1D-SIS-R$_{q,p,n\ell,\max\{n\cdot\ell\cdot B,T\}}$ on input $\boldsymbol{v} \in \mathbb{Z}_q^{n\ell}$:

1. Let $j \in \{0,1\}$ denote the first bit of x and set $\boldsymbol{w}^j := \boldsymbol{v} \in \mathbb{Z}_q^{n\ell}$.
2. Sample $\boldsymbol{w}^{\bar{j}} \leftarrow \mathbb{Z}_q^{n\ell}$
3. For $i = 0, \ldots, \ell - 1$:
   $(a_j)_i = \sum_{k=0}^{n-1} w^j_{in+k} X^k$
   $(a_{\bar{j}})_i = \sum_{k=0}^{n-1} w^{\bar{j}}_{in+k} X^k$
4. Run $r \leftarrow \mathcal{D}(x, \boldsymbol{a}_0, \boldsymbol{a}_1)$.
5. If there is no coefficient of $\boldsymbol{a}_x \cdot r$ in the set $(q/p) \cdot \mathbb{Z} + [-T', T']$, then abort.
6. Otherwise let x' be the input x with the first bit removed. There is a coefficient of $\boldsymbol{a}_x \cdot r = \boldsymbol{a}_j \cdot G^{-1}(\boldsymbol{a}_{x'}) \cdot r$ in $(q/p) \cdot \mathbb{Z} + [-T, T]$ meaning that for some k^*, there is a column of $G^{-1}(\boldsymbol{a}_{x'}) \cdot r$, say $\boldsymbol{y} \in R_q^\ell$ such that the X^{k^*} coefficient of $\langle \boldsymbol{a}_j, \boldsymbol{y} \rangle$ is in $(q/p) \cdot \mathbb{Z} + [-T, T]$.
7. Let $\mathbb{1}_{(\cdot)}$ be an indicator function. Noting that the coefficient of X^{k^*} of $\langle \boldsymbol{a}_j, \boldsymbol{y} \rangle$ is equal to
$$\sum_{i=0}^{\ell-1} \sum_{k=0}^{n-1} v_{in+k} \cdot (-1)^{\mathbb{1}_{k>k^*}} (y_i)_{k^*-k \bmod n},$$
output $\boldsymbol{z} \in \mathbb{Z}_q^{n\ell}$ where $z_{in+k} = (-1)^{\mathbb{1}_{k>k^*}} (y_i)_{k^*-k \bmod n}$ for $i = 0, \ldots, \ell - 1$, $k = 0, \ldots, n-1$.

It is clear that if $\mathcal{A}$ does not abort, it outputs a vector $\boldsymbol{z} \in \mathbb{Z}_q^{n\ell}$ such that $\langle \boldsymbol{v}, \boldsymbol{z} \rangle \in (q/p) \cdot \mathbb{Z} + [-T, T]$. Furthermore, if no abort occurs, then the entries of $\boldsymbol{z}$ (up to a sign) correspond to the coefficients of a column of $r \cdot G^{-1}(\boldsymbol{a}_{x'})$ where

$\|r\|_\infty \leq B$ with non-negligible probability. Recalling that $G^{-1}(\boldsymbol{a}_{x'}) \in R_q^{\ell \times \ell}$ is a binary decomposition of polynomials, we can see that,

$$||\boldsymbol{z}||_\infty \leq \ell \cdot n \cdot B$$

with non-negligible probability. In other words, $\mathcal{A}$ solves the $\mathsf{1D\text{-}SIS\text{-}R}_{q,p,n\ell,\max\{n\ell B,T\}}$ problem in polynomial time with non-negligible probability. To complete the proof, we use Lemma 11. □

C Instantiating Proof System 1

In this section, we describe in more detail how to instantiate our zero-knowledge arguments for Proof System 1 in terms of the protocol of Yang et al [58]. Let q be a prime-power, (n, m) be dimensions, and let P_n denote the power set of $\{1, \ldots, n\}^3$. The protocol of interest allows to prove knowledge of witnesses $\vec{x}$ to instances $(\boldsymbol{A}, \vec{y}, \mathcal{M})$ of the instance-witness set given by

$$\mathcal{R}^* = \left\{ \left((\boldsymbol{A}, \vec{y}, \mathcal{M}), \vec{x} \right) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m \times P_n \times \mathbb{Z}_q^n : \begin{smallmatrix} \boldsymbol{A} \cdot \vec{x} = \vec{y} \bmod q \ \wedge \\ \forall (i,j,k) \in \mathcal{M}, x_i = x_j \cdot x_k \bmod q \end{smallmatrix} \right\}.$$

Therefore, reducing our statement in Proof System 1 to the form of $\mathcal{R}^*$ suffices to describe an instantiation. Additionally, the Yang et al. protocol has a soundness error of $2/(2\bar{p}+1)$ per repetition where $\bar{p} = \mathsf{poly}(\kappa)$. Therefore, only $\kappa/\log(\bar{p})$ parallel repetitions are required to reach a soundness error of $2^{-\kappa}$.

The ZK Relation. Recall that $G^{-1} : R_q^{1 \times \ell} \to R_q^{\ell \times \ell}$ is the non-linear binary decomposition operation (on ring elements), and $G : R_q^{\ell \times \ell} \to R_q^{1 \times \ell}$ is the powers of two matrix that undoes G^{-1} i.e. $G := (1, 2, \ldots, 2^{\ell-1})$. Also recall that in the query phase, the client on input $x \in \{0,1\}^L$ computes the value $\boldsymbol{a}_x$ where

$$\boldsymbol{a}_x := \boldsymbol{a}_{x_0} \cdot G^{-1}\left(\boldsymbol{a}_{x_1} \cdot G^{-1}\left(\boldsymbol{a}_{x_2} G^{-1}\left(\ldots\right)\right)\right) \bmod q \tag{7}$$

using public $\boldsymbol{a}_0, \boldsymbol{a}_1$ and sends $\boldsymbol{c}_x = \boldsymbol{a} \cdot s + \boldsymbol{e}_1 + \boldsymbol{a}_x$. In terms of $\boldsymbol{a}_x$, the relation we are interested in providing a ZKPoK for is

$$\begin{aligned} \mathcal{R} = \{ \big((\boldsymbol{a}_0, \boldsymbol{a}_1, \boldsymbol{a}, \boldsymbol{c}_x), (s, x, \boldsymbol{e}_1) \big) \in (R_q^{1\times\ell})^4 \times \left(R, \{0,1\}^L, R^{1\times\ell}\right) \\ : \boldsymbol{c}_x = \boldsymbol{a} \cdot s + \boldsymbol{e}_1 + \boldsymbol{a}_x, \|s\|_\infty \leq \beta, \|\boldsymbol{e}_1\|_\infty \leq \beta \} \end{aligned}$$

where $\beta := \sigma \cdot \sqrt{n}$. To begin with, we can describe the computation of $\boldsymbol{a}_x$ recursively as well as $\boldsymbol{c}_x$ using

$$\begin{aligned} B_{L-1} &= G^{-1}(\boldsymbol{a}_{x_{L-1}}) \\ B_{L-2} &= G^{-1}(\boldsymbol{a}_{x_{L-2}} \cdot B_{L-1}) \\ B_{L-3} &= G^{-1}(\boldsymbol{a}_{x_{L-3}} \cdot B_{L-2}) \\ &\vdots \\ B_0 &= G^{-1}(\boldsymbol{a}_{x_0} \cdot B_1) \\ \boldsymbol{c}_x &= \boldsymbol{a} \cdot s + \boldsymbol{e}_1 + G \cdot B_0 \end{aligned}$$

where each equation is considered over the ring R_q. Importantly, $B_i \in R_2^{\ell \times \ell}$ represent binary decompositions and $\boldsymbol{a}_0$ and $\boldsymbol{a}_1$ are both in $R_q^{1 \times \ell}$.

Evaluation of F' as a System of Linear Equations. Unfortunately, the system of equations above is not linear due to the x_i and B_{i+1} terms and the fact that G^{-1} is not a linear operator. In the hope of deriving a linear system of equations, we first multiply by the linear operator $G \in R_q^{1 \times \ell}$ or equivalently $\boldsymbol{g}^T = (1, 2, \ldots, 2^{\ell-1}) \in R_q^{1 \times \ell}$. We also represent the terms $\boldsymbol{a}_{x_i} \cdot B_{i+1}$ as $\boldsymbol{a}_0 \cdot (1 - x_i) \cdot B_{i+1} + \boldsymbol{a}_1 \cdot x_i \cdot B_{i+1}$. In doing so, we can set $\boldsymbol{b}_0 = (\boldsymbol{g}^T \cdot B_0) \in R_q^{1 \times \ell}$ to obtain

$$
\begin{aligned}
\boldsymbol{g}^T \cdot B_{L-1} &= \boldsymbol{a}_0 \cdot (1 - x_{L-1}) + \boldsymbol{a}_1 \cdot x_{L-1} && (8)\\
\boldsymbol{g}^T \cdot B_{L-2} &= \boldsymbol{a}_0 \cdot (1 - x_{L-2}) \cdot B_{L-1} + \boldsymbol{a}_1 \cdot x_{L-2} \cdot B_{L-1} \\
\boldsymbol{g}^T \cdot B_{L-3} &= \boldsymbol{a}_0 \cdot (1 - x_{L-3}) \cdot B_{L-2} + \boldsymbol{a}_1 \cdot x_{L-3} \cdot B_{L-2} \\
&\vdots \\
\boldsymbol{b}_0 &= \boldsymbol{a}_0 \cdot (1 - x_0) \cdot B_1 + \boldsymbol{a}_1 \cdot x_0 \cdot B_1 && (9)\\
\boldsymbol{c}_x &= \boldsymbol{a} \cdot s + \boldsymbol{e}_1 + \boldsymbol{b}_0
\end{aligned}
$$

We now wish to come up with a ZKPoK allowing to prove knowledge of $\{(x_i)_{i=0}^{L-1}, (B_i)_{i=1}^{L-1}, \boldsymbol{b}_0, s, \boldsymbol{e}_1\}$ (where $s, \boldsymbol{e}_1$ are short, $x_i \in \{0, 1\}$ and $B_i \in R_2^{\ell \times \ell}$ are *binary* representations of ring elements whose coefficients are at most $q-1$) satisfying the above system of equations. Currently, the system is with respect to arithmetic in R_q rather than $\mathbb{Z}_q$. We now describe how to transform the above into a system over $\mathbb{Z}_q$. It should be clear that each ring multiplication can be represented by a matrix-vector multiplication over $\mathbb{Z}_q$. Taking note of this, we can define $\boldsymbol{A}_0, \boldsymbol{A}_1 \in \mathbb{Z}_q^{n \times n\ell}$ to be the matrices corresponding to multiplication by elements $\boldsymbol{a}_0, \boldsymbol{a}_1 \in R_q^{1 \times \ell}$ respectively. In particular, the j^{th} block of n columns of $\boldsymbol{A}_0$ (resp. $\boldsymbol{A}_1$) correspond to the negacyclic matrix representing multiplication by the j^{th} ring element of $\boldsymbol{a}_0$ (resp. $\boldsymbol{a}_1$). Additionally, denote by $\vec{a}_0$ and $\vec{a}_1 \in \mathbb{Z}_q^{n\ell}$, the coefficients of $\boldsymbol{a}_0$ and $\boldsymbol{a}_1$ respectively concatenated vertically. We next define $\boldsymbol{A} \in \mathbb{Z}_q^{n\ell \times n}$ to be the vertical concatenation of negacyclic matrices representing multiplication by the ring elements in $\boldsymbol{a}$. Then, we define $\vec{b}_i \in \mathbb{Z}_q^{n\ell^2}$ (for $i \in [L-1]$) to be a vector consisting of coefficients of the ring elements in B_i. In particular, the j^{th} block of $n\ell$ entries of $\vec{b}_i$ correspond to the coefficient vectors in the ring elements of the j^{th} column of B_i. Next we denote by $\vec{s}$, the vector of coefficients of the ring element s; by $\vec{c}$, the vertical concatenation of the coefficients of ring elements in $\boldsymbol{c}_x$; by $\vec{e}_1$, the vertical concatenation of the coefficients of ring elements in $\boldsymbol{e}_1$; and by $\vec{b}_0 \in \mathbb{Z}_q^{n\ell}$, the vector corresponding to the coefficients of ring elements in $\boldsymbol{b}_0$ concatenated vertically. Finally, defining $\boldsymbol{G}^{\otimes} = \boldsymbol{I}_\ell \otimes (\boldsymbol{g}^T \otimes \boldsymbol{I}_n)$ and $\boldsymbol{A}_i^{\otimes} = \boldsymbol{I}_\ell \otimes \boldsymbol{A}_i$ for $i \in \{1, 2\}$, we can rewrite the above

as the following system over $\mathbb{Z}_q$:

$$\begin{aligned}
\boldsymbol{G}^{\otimes} \cdot \vec{b}_{L-1} &= \vec{a}_0 \cdot (1 - x_{L-1}) + \vec{a}_1 \cdot x_{L-1} \\
\boldsymbol{G}^{\otimes} \cdot \vec{b}_{L-2} &= \boldsymbol{A}_0^{\otimes} \cdot (1 - x_{L-2}) \cdot \vec{b}_{L-1} + \boldsymbol{A}_1^{\otimes} \cdot x_{L-2} \cdot \vec{b}_{L-1} \\
\boldsymbol{G}^{\otimes} \cdot \vec{b}_{L-3} &= \boldsymbol{A}_0^{\otimes} \cdot (1 - x_{L-3}) \cdot \vec{b}_{L-2} + \boldsymbol{A}_1^{\otimes} \cdot x_{L-3} \cdot \vec{b}_{L-2} \\
&\vdots \\
\vec{b}_0 &= \boldsymbol{A}_0^{\otimes} \cdot (1 - x_0) \cdot \vec{b}_1 + \boldsymbol{A}_1^{\otimes} \cdot x_0 \cdot \vec{b}_1 \\
\vec{c} &= \boldsymbol{A} \cdot \vec{s} + \vec{e}_1 + \vec{b}_0
\end{aligned} \tag{10}$$

Let $\vec{1}_n$ (resp. $\vec{1}_{n\ell}$) denote the n-dimensional (resp. $n\ell$-dimensional) vector with all entries equal to 1. Now, to address the fact that the entries of $\vec{s}$ and $\vec{e}_1$ are upper bounded in absolute value by β, we consider the vectors $\vec{s}^{+} := \vec{s} + \beta\vec{1}_n$ and $\vec{e}_1^{+} := \vec{e}_1 + \beta\vec{1}_{n\ell}$ whose entries ought to lie in the set $\{0, 1, \ldots, 2\beta\}$. We also deploy a special form of binary decomposition. In particular, set $\delta_{2\beta,j} = \lfloor (2\beta + 2^{j-1})/2^j \rfloor$ for $j = 1, \ldots, \lfloor \log 2\beta \rfloor + 1$ and $\boldsymbol{D}_{2\beta} := \boldsymbol{I}_n \otimes (\delta_{2\beta,1}, \ldots, \delta_{2\beta,\lfloor \log 2\beta \rfloor + 1})$. As in [41], we can efficiently find a vector $\vec{s}' \in \{0,1\}^{n(\lfloor \log 2\beta \rfloor + 1)}$ such that $\boldsymbol{D}_{2\beta}\vec{s}' = \vec{s}^{+}$ for any $\vec{s}^{+} \in \{0, \ldots, 2\beta\}^n$. In addition, $\sum_{i=1}^{\lfloor \log 2\beta \rfloor + 1} \delta_i = 2\beta$ so for any $\vec{x}' \in \{0,1\}^{n(\lfloor \log 2\beta \rfloor + 1)}$, $\|\boldsymbol{D}_{2\beta} \cdot \vec{x}'\|_\infty \leq 2\beta$. We define $\vec{e}_1'$ analogously for the vector $\vec{e}_1^{+}$. Using these new variables, we can rewrite our system over $\mathbb{Z}_q$ as

$$\begin{aligned}
\boldsymbol{G}^{\otimes} \cdot \vec{b}_{L-1} &= \vec{a}_0 \cdot (1 - x_{L-1}) + \vec{a}_1 \cdot x_{L-1} \\
\boldsymbol{G}^{\otimes} \cdot \vec{b}_{L-2} &= \boldsymbol{A}_0^{\otimes} \cdot (1 - x_{L-2}) \cdot \vec{b}_{L-1} + \boldsymbol{A}_1^{\otimes} \cdot x_{L-2} \cdot \vec{b}_{L-1} \\
&\vdots \\
\boldsymbol{G}^{\otimes} \cdot \vec{b}_1 &= \boldsymbol{A}_0^{\otimes} \cdot (1 - x_1) \cdot \vec{b}_2 + \boldsymbol{A}_1^{\otimes} \cdot x_1 \cdot \vec{b}_2 \\
\vec{b}_0 &= \boldsymbol{A}_0^{\otimes} \cdot (1 - x_0) \cdot \vec{b}_1 + \boldsymbol{A}_1^{\otimes} \cdot x_0 \cdot \vec{b}_1 \\
\vec{c} &= \boldsymbol{A} \cdot \boldsymbol{D}_{2\beta} \cdot \vec{s}' + \boldsymbol{D}_{2\beta} \cdot \vec{e}_1' + \vec{b}_0 - \beta\left(\boldsymbol{A} \cdot \vec{1}_n + \vec{1}_{n\ell}\right)
\end{aligned} \tag{11}$$

The final modification ensures that $\vec{b}_i$ for $i \in \{1, \ldots, L-1\}$ really do correspond to binary decompositions of elements of R_q. Note that since q is not a power-of-two, one could satisfy the above linear system by using ℓ-bit binary decompositions of integers larger than q (as long as they are correct modulo q). Such a solution does not correspond to a correct PRF evaluation. To prevent this, we use $\boldsymbol{D}_{q-1} = \boldsymbol{I}_{n\ell} \otimes (\delta_{q,1}, \ldots, \delta_{q,\lfloor \log q-1 \rfloor + 1})$ and binary decompositions $\vec{h}_i$ such that $\vec{b}_i = \boldsymbol{D}_{q-1} \cdot \vec{h}_i$ for $i \in \{1, \ldots, L-1\}$. We are now ready to write out a system of the form considered by $\mathcal{R}^*$. The witness takes the form

$$\Big(\widetilde{x}_{L-1}, \widetilde{x}'_{L-1}, \ldots, \widetilde{x}_0, \widetilde{x}'_0, \widetilde{\vec{b}}_{L-1}, \ldots, \widetilde{\vec{b}}_1, \widetilde{\vec{b}}_0, \vec{f}_{L-1}, \vec{f}'_{L-1} \ldots, \vec{f}_1, \vec{f}'_1 \\ \widetilde{\vec{s}'}, \widetilde{\vec{e}'_1}, \widetilde{\vec{h}}_{L-1}, \ldots, \widetilde{\vec{h}}_1\Big)$$

where the dimensions and quadratic constraints are:

- For $i \in \{0, \ldots, L-1\}$, $\widetilde{x}_i, \widetilde{x}'_i \in \{0,1\}$ i.e. $\widetilde{x}_i = \widetilde{x}_i \cdot \widetilde{x}_i$
- For $i \in \{1, \ldots, L-1\}$, $\widetilde{\vec{b}}_i \in \{0,1\}^{n\ell^2}$ i.e. $\widetilde{\vec{b}}_i = \widetilde{\vec{b}}_i \odot \widetilde{\vec{b}}_i$ where $\odot$ is a component-wise product of vectors
- $\widetilde{\vec{b}}_0 \in \mathbb{Z}_q^{n\ell}$
- For $i \in \{1, \ldots, L-1\}$, $\vec{f}_i = \widetilde{x}_{i-1} \cdot \widetilde{\vec{b}}_i \in \mathbb{Z}_q^{n\ell^2}$
- For $i \in \{1, \ldots, L-1\}$, $\vec{f}'_i = \widetilde{x}'_{i-1} \cdot \widetilde{\vec{b}}_i \in \mathbb{Z}_q^{n\ell^2}$
- $\widetilde{\vec{s}'}, \widetilde{\vec{e}'}_1 \in \{0,1\}^{n(\lfloor \log 2\beta \rfloor + 1)}$, i.e. $\widetilde{\vec{s}'} = \widetilde{\vec{s}'} \odot \widetilde{\vec{s}'}$ and $\widetilde{\vec{e}'}_1 = \widetilde{\vec{e}'}_1 \odot \widetilde{\vec{e}'}_1$
- For $i \in \{1, \ldots, L-1\}$, $\widetilde{\vec{h}}_i \in \{0,1\}^{n\ell^2}$ i.e. $\widetilde{\vec{h}}_i = \widetilde{\vec{h}}_i \odot \widetilde{\vec{h}}_i$

Note that although parts of the witness will not appear explicitly in the linear system, they are required to ensure that we may define quadratic constraints to capture a correct PRF evaluation. The linear system over $\mathbb{Z}_q$ with respect to the constraints and witness above is:

$$
\begin{aligned}
1 &= \widetilde{x}_0 + \widetilde{x}'_0 \\
&\vdots \\
1 &= \widetilde{x}_{L-1} + \widetilde{x}'_{L-1} \\
\boldsymbol{G}^{\otimes} \cdot \widetilde{\vec{b}}_{L-1} &= \vec{a}_0 \cdot \widetilde{x}'_{L-1} + \vec{a}_1 \cdot \widetilde{x}_{L-1} \\
\boldsymbol{G}^{\otimes} \cdot \widetilde{\vec{b}}_{L-2} &= \boldsymbol{A}_0^{\otimes} \cdot \vec{f}'_{L-1} + \boldsymbol{A}_1^{\otimes} \cdot \vec{f}_{L-1} \\
&\vdots \\
\boldsymbol{G}^{\otimes} \cdot \widetilde{\vec{b}}_1 &= \boldsymbol{A}_0^{\otimes} \cdot \vec{f}'_2 + \boldsymbol{A}_1^{\otimes} \cdot \vec{f}_2 \\
\widetilde{\vec{b}}_0 &= \boldsymbol{A}_0^{\otimes} \cdot \vec{f}'_1 + \boldsymbol{A}_1^{\otimes} \cdot \vec{f}_1 \\
\vec{c} + \beta \left(\boldsymbol{A} \cdot \vec{1}_n + \vec{1}_{n\ell} \right) &= \boldsymbol{A} \cdot \boldsymbol{D}_{2\beta} \cdot \widetilde{\vec{s}'} + \boldsymbol{D}_{2\beta} \cdot \widetilde{\vec{e}'}_1 \\
\boldsymbol{D}_{q-1} \cdot \widetilde{\vec{h}}_1 &= \boldsymbol{G}^{\otimes} \cdot \widetilde{\vec{b}}_1 \\
&\vdots \\
\boldsymbol{D}_{q-1} \cdot \widetilde{\vec{h}}_{L-1} &= \boldsymbol{G}^{\otimes} \cdot \widetilde{\vec{b}}_{L-1}
\end{aligned}
$$

D An Alternative Construction and a Trick

Here, we will describe a protocol that appeared in a previous draft of this work. At a high level, this protocol can be seen as following a similar recipe to that of Figure 2 apart from that the client uses multiplicative blinding on $\boldsymbol{a}_x$ rather than additive blinding. In more detail, whereas our main construction emulates the DH blinding construction $H(x)^k = (H(x) \cdot g^r)^k / (g^k)^r$, this appendix aims to emulate the blinding construction $H(x)^k = \left((H(x)^r)^k \right)^{1/r}$. Specifically, it

emulates $H(x)^k = \left((H(x)^s)^k\right)^u \cdot \left(\left(H(x)^t\right)^k\right)^v$ for s, t, v, u s.t. $u \cdot s + v \cdot t = 1$. The main objective of this section is to outline a trick that may be applicable in other settings. To provide intuition for our alternative VOPRF design, we describe a basic protocol below that serves as a *starting point*.

1. The server publishes some commitment to a small key $k \in R_q$.
2. On input x, the client picks *invertible* $s \in R_q$, small $\boldsymbol{e} \in R_q^{1\times\ell}$ and sends $\boldsymbol{c}_x = \boldsymbol{a}_x \cdot s + \boldsymbol{e}$.
3. On input small $k \in R_q$, the server sends $\boldsymbol{d}_x = \boldsymbol{c}_x \cdot k + \boldsymbol{e}'$ for small $\boldsymbol{e}' \in R_q^{1\times\ell}$.
4. The client outputs $\boldsymbol{y} = \left\lfloor \frac{p}{q} \cdot \boldsymbol{d}_x \cdot s^{-1} \right\rceil$.

We will focus on correctness to motivate our trick while ignoring security. Note that we *would like to say* that

$$\left\lfloor \frac{p}{q} \cdot \boldsymbol{d}_x \cdot s^{-1} \right\rceil = \left\lfloor \frac{p}{q} \cdot \boldsymbol{a}_x \cdot k + \frac{p}{q}(\boldsymbol{e} \cdot k \cdot s^{-1} + \boldsymbol{e}' \cdot s^{-1}) \right\rceil = \left\lfloor \frac{p}{q} \cdot \boldsymbol{a}_x \cdot k \right\rceil.$$

Thus, we guarantee correctness if all coefficients of $\frac{p}{q} \cdot \boldsymbol{a}_x \cdot k$ are at least

$$\left| \frac{p}{q} \left(\boldsymbol{e} \cdot k \cdot s^{-1} + \boldsymbol{e}' \cdot s^{-1}\right) \right|_\infty$$

away from $\mathbb{Z} + \frac{1}{2}$. It turns out that if all coefficients of s^{-1} are small, then this condition is satisfied with extremely high probability due to the 1-dimensional short integer solution (1D-SIS) assumption [15]. As before, the form of $\boldsymbol{a}_x$ is crucial to the connection with the 1D-SIS problem. In particular, we rely on the fact that we can decompose $\boldsymbol{a}_x$ as $\boldsymbol{a}'_1 \cdot \boldsymbol{a}'_2$ where $\boldsymbol{a}'_1 \in R_q^{1\times\ell}$ is uniform random and $\boldsymbol{a}'_2 \in R_q^{\ell\times\ell}$ has entries that are polynomials with *binary* coefficients.

Unfortunately, this simplified protocol cannot quite be realised using standard RLWE secret distributions. The problem is that (to our knowledge) there is no standard RLWE secret distribution where samples from the distribution are guaranteed to have *small inverses* in R_q. To overcome this issue, we apply a technique for sampling "full" NTRU keys [32,52]. Firstly, we sample small ring elements s and t from a discrete Gaussian distribution. Secondly, we use the extended GCD algorithm – in combination with Babai's rounding algorithm – to recover small u and v, such that $u \cdot s + v \cdot t = 1 \bmod R_q$. To adapt the basic protocol to our actual protocol, the client sends $\boldsymbol{c}_x^1 = \boldsymbol{a}_x \cdot s + \boldsymbol{e}_1$, $\boldsymbol{c}_x^2 = \boldsymbol{a}_x \cdot t + \boldsymbol{e}_2$ and receives back

$$\boldsymbol{d}_x^1 = \boldsymbol{c}_x^1 \cdot k + \boldsymbol{e}'_1, \quad \boldsymbol{d}_x^2 = \boldsymbol{c}_x^2 \cdot k + \boldsymbol{e}'_2.$$

The final output is then $\left\lfloor \frac{p}{q} \left(u \cdot \boldsymbol{d}_x^1 + v \cdot \boldsymbol{d}_x^2\right) \right\rceil$. Correctness then relies on the fact that

$$u \cdot \boldsymbol{d}_x^1 + v \cdot \boldsymbol{d}_x^2 = \boldsymbol{a}_x \cdot \overbrace{(us + vt)}^{=1} \cdot k + u \cdot (\boldsymbol{e}_1 \cdot k + \boldsymbol{e}'_1) + v \cdot (\boldsymbol{e}_2 \cdot k + \boldsymbol{e}'_2)$$

and that the quantities $u, v, k, \boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}'_1, \boldsymbol{e}'_2$ are small. For completeness, we include the algorithm for sampling (u, v) on input (s, t) as well as a discussion bounding the size of u and v below.

In the following we use: $\mathrm{res}(\cdot, \cdot)$ to refer to the computation of the resultant of two polynomials; $\mathrm{xgcd}(\cdot, \cdot)$ to refer to the computation of the extended GCD of two integers; and $s^\star$ to refer to the conjugate of s in R. The sampling algorithm $\mathsf{fullINTRU}(s, t)$ runs the following steps.

1. Compute $r_s = \mathrm{res}(s, X^n + 1) \in \mathbb{Z}$ and $u' \in R$ s.t. $u' \cdot s = r_s$.
2. Compute $r_t = \mathrm{res}(t, X^n + 1) \in \mathbb{Z}$ and $v' \in R$ s.t. $v' \cdot t = r_t$.
3. Compute $r, u'', v'' = \mathrm{xgcd}(r_s, r_t)$. If $r \neq 1$: abort
4. Set $\bar{u} = u'' \cdot u' \in R$ and $\bar{v} = v'' \cdot v' \in R$.
5. Run Babai's inverting and rounding algorithm [3]:
   (a) Compute
   $$r = \left\lfloor \frac{\bar{v} \cdot s^\star - \bar{u} \cdot t^\star}{s \cdot s^\star + t \cdot t^\star} \right\rceil .$$
   (b) Update $(u,\ v) = (\bar{u} + r \cdot t,\ \bar{v} - r \cdot s) \in R^2$.
6. If $\|(u, v)\| > 2n\sigma$, output $\perp$. Else, return u, v.

Probability that $r \neq 1$. To show that the sampling algorithm does not abort (with overwhelming probability) in Step 3, we need to show that the ideal generated by s and t is the full ring R with noticeable probability. Recalling that $s, t \leftarrow D_{R,\sigma}$, Lemma 4.2 and 4.4 of [55] shows that this probability is at least $1 - O(1)$ (for large enough σ). This implies that with noticeable probability, the algorithm does not abort in Step 4.

Upper Bound on $\|(u, v)\|$. In order to show that the sampling algorithm does indeed output a pair (u, v), we analyse the probability that $\|(u, v)\| > 2n\sigma$ here. Our analysis follows [55]. Babai's rounding technique is an efficient way of obtaining a candidate solution to the closest vector problem (CVP). Given a lattice Λ with basis B (which need not be an invertible square matrix) and a target point $\boldsymbol{t}$ in the real span of B, Babai's rounding technique outputs the lattice vector $\boldsymbol{w} = B\lfloor (B^T B)^{-1} B^T \boldsymbol{t} \rceil$. The offset vector obtained can therefore be written as

$$\boldsymbol{t} - \boldsymbol{w} = B \cdot \left((B^T B)^{-1} B^T \boldsymbol{t} - \lfloor (B^T B)^{-1} B^T \boldsymbol{t} \rceil \right) \in B \cdot \left[-\frac{1}{2}, \frac{1}{2} \right]^n . \tag{12}$$

Let $\boldsymbol{b}_i$ denote the i^{th} column of B for $i = 1, \ldots, n_B$. From Equation (12),

$$\|\boldsymbol{t} - \boldsymbol{w}\| \leq \sqrt{\sum_{i=1}^{n_B} (\|\boldsymbol{b}_i\|/2)^2} \leq \frac{\sqrt{n}}{2} \max_i \|\boldsymbol{b}_i\|. \tag{13}$$

We now use this analysis to give an upper bound on $\|(u, v)\|$ computed by $\mathsf{fullINTRU}(s, t)$. At a high level, the first four steps find a (potentially very long)

pair $(\bar{u}, \bar{v}) \in R^2$ such that $\bar{u} \cdot s + \bar{v} \cdot t = 1$ and the final two steps update this to (u, v) using Babai's rounding technique. In particular, suppose we define $S, T \in \mathbb{Z}^{n \times n}$ to be the negacyclic matrices denoting multiplication by s and t respectively. Then the final two steps run Babai's rounding technique on the lattice $\Lambda = \{\boldsymbol{z} \in \mathbb{Z}^{2n} : [S|T] \cdot \boldsymbol{z} = 0\}$ with basis $B = [T|-S]^T \in \mathbb{Z}^{2n \times n}$ (which has linearly independent columns by invertibility of s, t in the field $\mathbb{Q}(X)/\langle X^n + 1 \rangle$). Decomposing $(\bar{u}, \bar{v}) = (\bar{u}^{\perp}, \bar{v}^{\perp}) + (\bar{u}^{B}, \bar{v}^{B})$ where $(\bar{u}^{\perp}, \bar{v}^{\perp})$ is the component of $(\bar{u}, \bar{v})$ orthogonal to the span of B and $(\bar{u}^{B}, \bar{v}^{B})$ belongs to the span of B, we use the target vector $(\bar{u}^{B}, \bar{v}^{B})$. Bounding the norm of the offset (via Equation (13)) and noting that $(\bar{u}^{\perp}, \bar{v}^{\perp})$ is orthogonal to this offset gives us a bound for the final value of $\|(u, v)\|$. Each column of our basis consists of the coefficients of $s, t \leftarrow \chi_{\sigma}$, so by Lemma 1 we obtain the bound

$$\begin{aligned} \|(u, v)\| &\leq \|(\bar{u}^{\perp}, \bar{v}^{\perp})\| + \frac{\sqrt{n}}{2} \cdot \|(s, t)\| \\ &\leq \|(\bar{u}^{\perp}, \bar{v}^{\perp})\| + \frac{\sqrt{n}}{2} \cdot 2\sigma\sqrt{n} \\ &= \|(\bar{u}^{\perp}, \bar{v}^{\perp})\| + n\sigma \end{aligned}$$

that holds with all but negligible probability over the choice of s and t. To bound the above further, note that by definition we have $\|(\bar{u}^{\perp}, \bar{v}^{\perp})\| \leq \min_{r' \in K} \|(u + r't, v - r's)\|$. On considering $r' = vs^{-1}$ and the fact that $\bar{u}s + \bar{v}t = 1$, we have that $\|(\bar{u}^{\perp}, \bar{v}^{\perp})\| \leq \|(s^{-1}, 0)\|$. Lemma 4.1 of [55] shows that $\|s^{-1}\| \geq \omega(\sqrt{n})/\sigma$ only with probability $o(1)$. Therefore, we have that the pair (u, v) computed by $\mathsf{fullNTRU}(s, t)$ satisfies

$$\|(u, v)\| \leq n\sigma + n\sigma$$

with probability $1 - o(1)$, meaning that the sampling algorithm succeeds with noticeable probability.

E A Generic MPC Based Construction

One could alternatively instantiate VOPRFs using generic techniques for establishing Multi-Party Computation (MPC) protocols by treating a single execution of the VOPRF protocol, for a PRF like AES, as a single invocation of a classical two-party *actively secure* MPC protocol. Such protocols are now very efficient, and can be based on post-quantum assumptions. However, to match our requirements – the round-optimality of the scheme that we devise in this work – we would require that the MPC protocol can be completed in two rounds (one message from the client and one message from the server).

This essentially rules out approaches based on efficient LSSS-based protocols such as SPDZ [20]. In addition, in spite of recent advances, the SPDZ protocol [54] still requires RLWE parameter settings that use a similar size polynomial degree ($n \approx 2^{14}$), but with a more structured, and larger, modulus. Concretely, the construction of [54] requires that q has two prime factors with minimum sizes $p_0 \sim 2^{216}$ and $p_1 \sim 2^{164}$; our construction requires an unstructured q of size 2^{256}.

As such, any implementation using the SPDZ paradigm would be noticeably less efficient than our protocol. See Section 5.3 for more details on our parameters.

Due to the above remarks, the need for round-optimality leads us to consider Garbled Circuit based constructions; where the server (say) creates a garbling of the PRF which is then evaluated by the client. A VOPRF supports being called many times by many different callers, while some applications might not require it, the primitive supports it. Thus a possible solution would be for the initialization phase to consist of the server choosing a key k for the AES cipher, and then committing to it by computing and publishing $h = H(k\|r)$ for some random value r and some hash function H (e.g. SHA-256 or SHA-3). Then on invocation by a client for some input value x, the server and the client could engage in a two-party protocol to compute the function, where we denote by $[v]_s$ (and $[v]_c$) a value which is kept secret by the server (resp. the client), and $[v]$ a value which is hidden from both the client and the server.

- $[v] \leftarrow \mathsf{AES}([k]_s, [x]_c)$.
- $[b] \leftarrow (h \neq H([k]_s, [r]_s))$.
- $[a]_c \leftarrow ([b] \cdot \perp) \; + \; (1 - [b]) \cdot [v]$.

Note, the need for the checking of the public commitment h to avoid the server changing the key on each invocation, or having a different key for different clients.

To obtain our security guarantees we would need the protocol to be executed using an *actively secure* two-party protocol. But such a protocol is impossible to realise using our requisite two rounds. Indeed actively secure two-party computation requires a minimum of five rounds (see [37]). If one was only interested in passive security, then a four round protocol would follow from Yao's original two-party construction. Thus whilst the *execution time* for a Garbled-Circuit based construction may be more efficient than our protocol, the round complexity will never match our protocol. It is thus an open problem to obtain a VOPRF protocol with the efficiency of (say) the passively secure implementation via Yao's protocol of the above function, with the round complexity of our own lattice based construction.

F Proof-of-concept Implementation

```
"""
A very rough(!) proof of concept implementation.
"""

from sage.all import matrix, vector
from sage.all import ceil, log, xgcd, next_prime, inverse_mod
from sage.all import ZZ, QQ, IntegerModRing


class VOPRFPoC(object):
    """
    Figure 2
    """

    def __init__(self, n, p, q):
        """
```

```
        Setup

        :param n: ring dimension, must be power of two
        :param p: rounding modulus
        :param q: computation modulus

        """
        self.q = q
        self.p = p
        self.n = n
        self.ell = ceil(log(q, 2))

        # NOTE: we do modular reductions mod phi = X^n+1 hand
        self.R_Z = ZZ["x"]
        self.X = self.R_Z.gen()
        self.phi = self.X ** n + 1
        self.R = IntegerModRing(q)["x"]

        self.a = [
            vector(
                self.R,
                self.ell,
                [
                    self.R.random_element(degree=n - 1)
                    for _ in range(self.ell)
                ],
            )
            for _ in range(3)
        ]

        self.k = self.sample_small()
        self.c = self.a[2] * self.k + self.sample_small(
            scalar=False
        )  # a[2] == a in Figure 2

    def bp14(self, x, k=1):
        """
        Evaluate BP14 on input `x` and key `k`

        :param x: a vector of bits
        :param k: a small element in R

        """
        L = len(x)
        t = 1
        for i in range(1, L)[::-1]:
            t = self.ginv(self.modred(self.a[x[i]] * t))
        ax = self.modred(self.a[x[0]] * t)

        e = self.sample_small(scalar=False)
        return self.modred(ax * k) + e

    def sample_small(self, bound=1, scalar=True):
        """
        Sample a small element in R or R^l

        :param bound: l_oo bound on coefficients
        :param scalar: if `True` return element in R, otherwise in R^l

        """
        if scalar:
            return self.R_Z.random_element(
                degree=self.n - 1, x=-bound, y=bound + 1
            )
        else:
            return vector(
                self.R_Z,
                self.ell,
```

```
            [self.sample_small() for _ in range(self.ell)],
        )

    def modred(self, v):
        """
        Reduce an element in R^l modulo phi

        :param v: an element in R^l

        """
        return vector(self.R, self.ell, [v_ % self.phi for v_ in v])

    def nice_lift(self, h):
        """
        Return a balanced integer representation of an element in R.

        :param h: an element in R

        """
        r = []
        h = h % self.phi
        for e in h:
            if ZZ(e) > self.q // 2:
                r.append(ZZ(e) - self.q)
            else:
                r.append(ZZ(e))
        return self.R_Z(r)

    def ginv(self, a):
        """
        Return G^-1(a), i.e. bit decomposition.

        :param a: an element in R^l

        """

        A = matrix(self.R, self.ell, self.ell)
        for i in range(self.ell):
            a_ = a[i].change_ring(ZZ)
            for j in range(self.ell):
                A[j, i] = self.R(
                    [(ZZ(a__) // 2 ** j) % 2 for a__ in a_]
                )
        assert self.G * A == a
        return A

    @property
    def G(self):
        """
        Vector G = [1,2,4,...] in R^l
        """
        return vector(
            self.R, self.ell, [2 ** i for i in range(self.ell)]
        )

    def __call__(self, x):
        """
        Run the protocol on ‘x‘, ignoring zero-knowledge proofs

        :param x: a vector of bits

        """
        # CLIENT

        s = self.sample_small(scalar=True)
        e1 = self.sample_small(scalar=False)
        cx = self.bp14(x) + self.a[2] * s + e1
```

```
        # SERVER

        e_ = self.sample_small(bound=2 ** 64, scalar=False)
        dx = self.modred(cx * self.k + e_)
        # CLIENT

        y = self.nice_lift((dx - self.c * s)[0])
        return y // (self.q / self.p)


class AltVOPRFPoC(VOPRFPoC):
    """
    """

    def __init__(self, n, p, q):
        """
        Setup

        :param n: ring dimension, must be power of two
        :param p: rounding modulus
        :param q: computation modulus

        """
        self.q = q
        self.p = p
        self.n = n
        self.ell = ceil(log(q, 2))

        # NOTE: we do modular reductions mod phi = X^n+1 hand
        self.R_Z = ZZ["x"]
        self.X = self.R_Z.gen()
        self.phi = self.X ** n + 1
        self.R = IntegerModRing(q)["x"]

        self.a = [
            vector(
                self.R,
                self.ell,
                [
                    self.R.random_element(degree=n - 1)
                    for _ in range(self.ell)
                ],
            )
            for _ in range(2)
        ]

        self.k = self.sample_small()

    def full_ntru(self, s, t):
        """
        Return small `u,v` s.t. `u*s + v*t = 1`

        :param s: a small element in R
        :param t: a small element in R

        """
        Rs = s.resultant(self.phi)
        Rt = t.resultant(self.phi)
        u_ = (Rs * s.change_ring(QQ).inverse_mod(self.phi)) % self.phi
        v_ = (Rt * t.change_ring(QQ).inverse_mod(self.phi)) % self.phi
        r, u__, v__ = xgcd(Rs, Rt)
        u = u__ * u_
        v = v__ * v_
        u = u.change_ring(ZZ)
        v = v.change_ring(ZZ)

        def conjugate(f):
            ft = -f[self.n // 2] * self.X ** (self.n // 2) + f[0]
```

```
            for i in range(1, self.n // 2):
                ft += (
                    -f[i] * self.X ** (self.n - i)
                    - f[self.n - i] * self.X ** i
                )
            return ft

        def xgcd_reduce(f, g, G, F):
            """
            https://eprint.iacr.org/2019/015 solves f*G' - g*F' == f*G - g*F

            We map s,t,u,v to f,g, -G, F.

            """
            f, g, F, G = f, g, F, -G
            for j in range(32):
                num = (F * conjugate(f) + G * conjugate(g)) % self.phi
                den = (f * conjugate(f) + g * conjugate(g)) % self.phi
                k = (
                    num
                    * inverse_mod(den.change_ring(QQ), self.phi)
                    % self.phi
                )
                k = sum(
                    [
                        round(c) * self.X ** i
                        for i, c in enumerate(list(k))
                    ]
                )
                if k == 0:
                    break
                F, G = (F - k * f) % self.phi, (G - k * g) % self.phi
            return -G, F

        u, v = xgcd_reduce(s, t, u, v)

        return u, v

    def __call__(self, x):
        """
        Run the protocol on `x`, ignoring zero-knowledge proofs

        :param x: a vector of bits

        """
        # CLIENT

        while True:
            s = self.sample_small()
            t = self.sample_small()
            u, v = self.full_ntru(s, t)
            if (u * s + v * t) % self.phi == 1:
                break

        c1 = self.bp14(x, s)
        c2 = self.bp14(x, t)

        # SERVER

        d1 = self.modred(
            c1 * self.k + self.sample_small(bound=2 ** 64, scalar=False)
        )  # for "drowning"
        d2 = self.modred(
            c2 * self.k + self.sample_small(bound=2 ** 64, scalar=False)
        )

        # CLIENT
```

```
        yx = self.nice_lift((u * d1 + v * d2)[0])
        return yx // (self.q / self.p)

# instantiate with some toy parameters

def test(cls=VOPRFPoC, p=3, q_size=96, n=256):
    q_ = next_prime(2 ** q_size)
    voprf = cls(n, p, p * q_)
    return voprf([0, 1]) == voprf([0, 1])
```