Practical Post-Quantum Few-Time Verifiable Random Function with Applications to Algorand

Muhammed F. Esgin[1,2], Veronika Kuchta[3], Amin Sakzad[1], Ron Steinfeld[1], Zhenfei Zhang[4], Shifeng Sun[1], and Shumo Chu[5]

[1] Monash University, Australia
[2] Data61, CSIRO, Australia
[3] The University of Queensland, Australia
[4] Manta Network**
[5] University of California, Santa Barbara

Abstract. In this work, we introduce the *first practical post-quantum* verifiable random function (VRF) that relies on well-known (module) lattice problems, namely Module-SIS and Module-LWE. Our construction, named LB-VRF, results in a VRF value of only 84 bytes and a proof of around only 5 KB (in comparison to several MBs in earlier works), and runs in about 3 ms for evaluation and about 1 ms for verification.
In order to design a practical scheme, we need to restrict the number of VRF outputs per key pair, which makes our construction *few-time*. Despite this restriction, we show how our few-time LB-VRF can be used in practice and, in particular, we estimate the performance of Algorand using LB-VRF. We find that, due to the significant increase in the communication size in comparison to classical constructions, which is inherent in all existing lattice-based schemes, the throughput in LB-VRF-based consensus protocol is reduced, but remains practical. In particular, in a medium-sized network with 100 nodes, our platform records a 1.14× to 3.4× reduction in throughput, depending on the accompanying signature used. In the case of a large network with 500 nodes, we can still maintain at least 24 transactions per second. This is still much better than Bitcoin, which processes only about 5 transactions per second.

Keywords: Post-Quantum · Verifiable Random Function · Blockchain · Lattice · Algorand

1 Introduction

The notion of verifiable random function (VRF) was put forth by Micali, Rabin and Vadhan [39]. It allows a user to generate a random value that is both authenticated and publicly verifiable. VRFs have been used in practice, for example, in DNSSEC protocol [26], and in blockchain consensus protocols [25,13] to establish Proof-of-Stake. In both cases, a VRF serves

** Work was done while with Algorand.

as a fundamental building block to provide verifiable random inputs to the protocol. There are currently two main VRF constructions namely, ECVRF [42] (based on elliptic curves), and BLS-VRF [9,8] (based on pairings). Specifically, ECVRF over curve25519 is currently in the standardization process by CRFG [27] and is deployed by Algorand [25,13], while BLS-VRF is adopted by Dfinity [30].

The main drawback of the above-mentioned VRF constructions is that they are vulnerable to quantum attacks. This is a significant concern especially in the blockchain setting since attackers may "rewrite history" if they are able to forge the VRF (with a quantum computer). Let us explain why this is a major concern even today. In a blockchain use case as in Algorand, VRF is used to ensure that the committee members are selected honestly for all the blocks that are already committed on the chain. A new user, who has no record of the previous blocks, can be assured of the validity of the blocks by looking at the votes that has been recorded as long as VRF remains secure. In such protocols, since a block is agreed by the majority of the committee members, there will never be a fork of the blockchain. However, when the VRF security is compromised, one can "rewrite history" by corrupting selected committee members for any given round (including rounds in the past), and then can create a fork to the blockchain subsequent to that round. As a result, a potential future security threat against the integrity of VRFs is important even today. To circumvent such a threat, in this paper we introduce the *first post-quantum* VRF construction that does not rely on heavy machinery and meets practical efficiency levels. We emphasize that our focus in this paper is realization of *practical* constructions.

Technical challenges in the lattice setting. Construction of an *efficient* lattice-based VRF is quite challenging as realizing long-term pseudorandomness and uniqueness properties together (while maintaining practicality) does not go well in the lattice setting. To understand why that is the case, let us first briefly explain how ECVRF works.

In ECVRF, the secret key is a field element x and the corresponding public key is a group element xG for some public generator G. The ECVRF output is then a group element xP, where $P = H(xG, \mu)$ is computed deterministically from the VRF input μ and a public key xG for some publicly computable function H. Then, a sigma protocol (with Fiat-Shamir transformation) is applied to prove that both the VRF output xP and the public key xG have the same discrete logarithm with respect to P and G, respectively (i.e., $x = \log_G(xG) = \log_P(xP)$). In essence, pseudorandomness follows from DDH assumption and the uniqueness comes

from the fact that, for a fixed input μ and a fixed public key xG, there is a unique xP such that $x = \log_G(xG)$ and $P = H(xG, \mu)$.

Issue with long-term pseudorandomness in the lattice setting. The main technique to hide a secret key $\mathbf{s}$ in lattice-based cryptography is to disturb a lattice point by computing $t = \langle \mathbf{b}, \mathbf{s} \rangle + e$, where $\mathbf{b}$ is a public vector, $\mathbf{s}$ is the secret vector and e is a *small* (secret) error sampled from some error distribution. Assuming that computations are done over a ring $\mathfrak{R}$, t is precisely a Module-LWE (MLWE) sample in $\mathfrak{R}$ and is indistinguishable from a uniformly random element in $\mathfrak{R}$ based on MLWE.

Now, let's see the difficulty in constructing an MLWE-based lattice analogue of the above DDH-based VRF. In this lattice-based VRF, for a fixed user secret key $\mathbf{s}$, one can map an input message μ together with the user public key to a vector $\mathbf{b} = H(pk, \mu)$ using a deterministic function H (modelled as a random oracle). From here, with the hope of hiding the secret $\mathbf{s}$, one may attempt to compute the corresponding VRF value as $v = \langle \mathbf{b}, \mathbf{s} \rangle + e$ for some error e sampled from an error set of many elements. However, unlike the DDH-based setting above, this approach violates the uniqueness property as there are multiple small e values that can be used, and thus multiple possible VRF values for a given (pk, μ).

An alternative approach could be to choose the error in a *deterministic* way. In particular, one may compute $v = \texttt{Round}(\langle \mathbf{b}, \mathbf{s} \rangle)$ for some rounding function $\texttt{Round}(\cdot)$, which simply chops off some least significant bits, and rely for pseudorandomness on the learning with rounding (LWR) problem [5]. In fact, this approach has been used to construct lattice-based pseudorandom functions (PRFs) [35,45]. The issue here is that currently, there is no known *efficient* zero-knowledge proof to prove that the VRF evaluator indeed computed v in this fashion. For example, the recent results from [45] yield such a proof of size in the order of several MBs. Therefore, this approach does not address our practical goals.

Issue with uniqueness in the lattice setting. Another orthogonal issue is in relation to uniqueness. Efficient lattice-based signature schemes are non-deterministic and therefore standard transformation from a unique signature to a VRF (as given in [39]) does not trivially work. Moreover, the approach taken in [42] to prove uniqueness of the ECVRF construction also does not apply in the lattice setting. In particular, the authors in [42] show that for any given VRF output that is not generated honestly and any valid proof, there exists a *single* random oracle output c that can make the proof verify. As the chance of hitting that challenge is negligibly small, the uniqueness follows. However, the same idea does not work in the lattice setting.

1.1 Our Contribution

We propose the *first practical* verifiable random function, named LB-VRF, based on standard post-quantum hardness assumptions, namely Module-SIS (MSIS) and Module-LWE (MLWE). A single LB-VRF proof costs around 5KB and runs in about 3 ms for evaluation and 1 ms for verification. To show the practicality of our results, we implemented LB-VRF and tested it in practice. We discuss the implementation and evaluation further below.

The main drawback of our construction is that a single key pair can only be used to generate a limited number k of VRF outputs. Therefore, we say that our LB-VRF construction is 'k-time'. However, we show that this aspect is not a significant disadvantage in the blockchain setting as the users can frequently update their keys. In fact, some privacy-enhanced blockchain applications such as Monero and Zcash employ one-time public keys per transaction (see, for example, [44,41,43,46,40,6,22]). For instance, as detailed in Section 4.1.6 of Zcash specification [32], a fresh signature key pair is generated for each transaction.

We also note that the aspect of being k-time is only required to satisfy *pseudorandomness* (i.e., to prevent the user secret key from being leaked), and is not related to the soundness (i.e., uniqueness). That is, it is infeasible for a cheating prover, even by violating the k-time property, to produce incorrect VRF outputs that pass the verification algorithm.

Main idea. A user secret key in LB-VRF is a short vector $\mathbf{s}$, and the public key becomes $\mathbf{t} = \mathbf{A}\mathbf{s}$ for a public matrix $\mathbf{A}$. Then, we use the so-called "Fiat-Shamir with Aborts" technique [37] to prove knowledge of the secret key. However, this proof is *relaxed* in the sense that it only proves knowledge of $\mathbf{s}'$ such that $\bar{c}\mathbf{t} = \mathbf{A}\mathbf{s}'$ for some secret relaxation factor $\bar{c}$ (i.e., the proof has a *knowledge gap*). This relaxation complicates the uniqueness proof. If we would want to prove an *exact* relation, then such a proof alone would require about 50 KBs [20], which we consider too costly for our target blockchain application.

From the discussion about the pseudorandomness in the introduction, the option that remains at hand, and the one we employ in LB-VRF for the computation of the VRF value v, is to use no error at all, i.e. set $v = \langle \mathbf{b}, \mathbf{s} \rangle$. This method only leaks a limited amount information on $\mathbf{s}$ for a relatively small number k of VRF outputs, but fortunately it suffices for our application of VRF to blockchain protocols. This method does, however, leak too much information on the secret $\mathbf{s}$ when *many* VRF outputs are computed with the same key. In particular, one cannot output, say, 2^{64} VRF values $v_i = \langle \mathbf{b}_i, \mathbf{s} \rangle$ where $\mathbf{b}_i = H(pk, \mu_i)$ (at least while still pre-

	ECVRF[4]	BLS-VRF[3]	LB-VRF
PK size	32 bytes	96 bytes	3.32 KB
Proof size	80 bytes	48 bytes	4.94 KB
Prove time	0.2 ms	0.6 ms	3.1 ms
Verification time	0.2 ms	2.0 ms	1.3 ms

Table 1. Comparison of our scheme and classical VRFs.

serving practicality). This issue with long-term pseudorandomness does not seem to be efficiently addressable with the existing lattice-based tools.

More concretely, we map the VRF input μ and the user public key $\mathbf{t}$ to a vector $\mathbf{b}$ using a random oracle. We then prove in zero-knowledge that the VRF value computed as $v = \langle \mathbf{b}, \mathbf{s} \rangle$ is well-formed. However, again due to the *relaxed* nature of the underlying zero-knowledge proof that we use to achieve short proofs, the uniqueness does not immediately follow. To handle this, we show via a "double rewinding" argument that as long as the MSIS problem is hard (with certain parameters), any two VRF outputs computed by an *efficient* uniqueness attack algorithm under the same public key and input must be the same (see the proof of Theorem 3.1). Therefore, we can only achieve *computational* uniqueness, based on the standard MSIS hardness assumption. In regards to pseudorandomness, we show that it follows from MLWE as long as at most k VRF outputs are produced under a single key pair.

To further reduce the VRF value size and increase computational efficiency, we introduce an additional optimization technique which performs the VRF value computation in a *subring* of a commonly-used cyclotomic ring. We show that the uniqueness security property is still preserved even when using this optimisation technique. This technique results in $\approx 8\times$ smaller VRF values for typical parameters compared to outputting the full ring element as the VRF output, and approximately doubles the evaluation and verification speed.

Implementation and deployment. We present an efficient implementation of our k-time LB-VRF. In particular, we implement the "worst-case" (in terms of performance) setting where a single key pair is used only once (i.e., $k = 1$) and show that even that case is practical. Our code is open-sourced[6]. We compare the performance of our scheme against ECVRF over curve25519 and BLS-VRF over BLS12-381 curve. The implementation details are provided in Section 4.

[6] `https://github.com/zhenfeizhang/lb-vrf`

VRF Type Sign. Type	ECVRF + Ed25519	LB-VRF + Ed25519	LB-VRF + Dilithium	LB-VRF + Falcon	LB-VRF + Rainbow
10 nodes	1000	1000	353	624	997
100 nodes	1000	862	292	532	860
500 nodes	1000	250	24	120	250
Assumption	ECC	lattice + ECC	lattice	lattice	lattice + MQ

Table 2. Performance comparison in terms of TPS (the numbers are approximate). TPS (transactions per second) is a generic metric used by multiple blockchain platforms. In comparison, Bitcoin achieves about 5 TPS.

Since our construction increases sizes significantly, it is important to understand how practical our scheme can really be in real world protocols. For a fair comparison, we also investigate the impact of integrating our scheme to the Algorand protocol. With both ECVRF and Ed25519 signatures, Algorand blockchain is able to transmit 5 MB of data per block, with a block generation time of less than 5 seconds. This allows Algorand to achieve 1000 transactions per second (TPS), with over 1000 nodes, as of today. We report the performance estimation of our LB-VRF with four different signatures, Ed25519 (used by Algorand), and 3 NIST PQC third round candidates. The data is presented in Table 2, and more details are provided in Section 4.

1.2 Related Work

Originally introduced by Micali, Rabin and Vadhan [39], VRFs have become an important cryptographic primitive in several applications. In [39], the authors show a relation between VRFs and unique signatures by combining the unpredictability property of a unique signature with the verifiability by extending the Goldreich-Goldwasser-Micali construction of a pseudorandom function [28]. The concept of a VRF has been investigated further in [36] and [17]. In [36] the authors provide a construction of a verifiable unpredictable function (VUF) from a unique signature scheme and turn it into a VRF using the original transform from [39]. The aforementioned VRFs are constructed from number-theoretic assumptions. More number-theoretic constructions are given in [16,31,33,1,7]. In [10] the authors introduced the notion of weak VRF where pseudorandomness is required to hold only for randomly selected inputs. Further VRF-related primitives such as simulatable VRF, constrained VRF have been introduced in [12,23].

On the side of quantum-safe proposals, feasibility of a lattice-based VRF was given in [29,45]. The construction in [29] relies on heavy ma-

chinery such as constrained PRFs and there is no practical efficiency evaluation provided. In the latter work, the authors in [45] briefly mention in a remark (without a rigorous security or performance analysis) that their zero-knowledge proofs give rise to a lattice-based VRF construction. However, the authors claim that this construction satisfies only *trusted* uniqueness, which is not sufficient for blockchain applications. Moreover, this construction is expected to be far from practical as even more basic proofs in [45] require MBs of communication.

2 Preliminaries

We use λ and to denote the security parameter a function negligible in λ. We define the polynomial rings $\mathcal{R} := \mathbb{Z}[x]/(x^d+1)$ and $\mathcal{R}_q := \mathbb{Z}_q[x]/(x^d+1)$ for d a power of 2. We denote by bold, capital letters (e.g. $\mathbf{M}$) matrices whose elements are in $\mathcal{R}$ and denote by bold, lower case letters (e.g. $\mathbf{v}$), vectors whose elements are in $\mathcal{R}$. $\mathbb{S}_c$ denotes the set of polynomials in $\mathcal{R}$ with infinity norm at most $c \in \mathbb{Z}^+$. We write 0^n to denote the n-dimensional zero vector and $\mathbf{I}_n$ for the n-dimensional identity matrix.

Let $\mathcal{R}_p \cong \mathcal{R}_p^{(1)} \times \cdots \times \mathcal{R}_p^{(s)}$ for some $s \geq 1$. That is, $\mathcal{R}_p^{(i)} = \mathbb{Z}_p[x]/(f_i(x))$ such that f_i with $\deg(f_i) = d/s$ is an irreducible factor of $x^d + 1 \bmod p$ for each $i = 1, \ldots, s$. In our LB-VRF construction, a set of operations will be performed in $\mathcal{R}_p^{(1)}$ for better efficiency. We will denote this ring by $\bar{\mathcal{R}}_p = \mathbb{Z}_p[x]/(f(x))$ (see Table 3 for the concrete ring $\bar{\mathcal{R}}_p$). The other subrings $\mathcal{R}_p^{(2)}, \ldots, \mathcal{R}_p^{(s)}$ of $\mathcal{R}_p$ will not be of concern for our construction.

Definition 2.1 ($\mathsf{MSIS}_{q,n,m,\beta}$ **[34]).** *Let $\mathcal{R}$ be some ring and $\mathcal{K}$ a uniform distribution over $\mathcal{R}_q^{n\times m}$. Given a random matrix $\mathbf{A} \in \mathcal{R}_q^{n\times m}$ sampled from $\mathcal{K}$, find a non-zero vector $\mathbf{v} \in \mathcal{R}_q^m$ such that $\mathbf{A} \cdot \mathbf{v} = \mathbf{0}$ and $\|\mathbf{v}\| \leq \beta$.*

Definition 2.2 ($\mathsf{MLWE}_{q,n,m,\chi}$ **[34]).** *Let χ be a distribution over $\mathcal{R}_q$, $\mathbf{s} \xleftarrow{\$} \chi^m$ be a secret key. The $\mathsf{MLWE}_{q,\mathbf{s}}$ distribution is obtained by sampling $\mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{n\times m}$ and error $\mathbf{e} \xleftarrow{\$} \chi^n$ and outputting $(\mathbf{A}, \mathbf{A}\cdot\mathbf{s} + \mathbf{e})$. The goal is to distinguish the $\mathsf{MLWE}_{q,\mathbf{s}}$ output from the uniform distribution $\mathcal{U}(\mathcal{R}_q^{n\times m}, \mathcal{R}_q^n)$.*

In our analysis, we use the following result that helps us argue the invertibility of challenge differences.

Lemma 2.3 ([38]). *Let $n \geq k > 1$ be powers of 2 and $p \equiv 2k + 1 \pmod{4k}$ be a prime. Any f in $\mathbb{Z}_p[X]/(X^n+1)$ is invertible if one of the following is satisfied*

$$0 < \|f\|_\infty < p^{1/k}/\sqrt{k} \quad \text{or} \quad 0 < \|f\| < p^{1/k}.$$

2.1 Verifiable Random Function

Definition 2.4 (Verifiable Random Function [39]). *Let* ParamGen, KeyGen, VRFEval, Verify *be polynomial-time algorithms where:*

$\mathsf{ParamGen}(1^\lambda)$**:** *On input a security parameter* 1^λ*, this probabilistic algorithm outputs some global, public parameter* pp*.*

$\mathsf{KeyGen}(pp)$**:** *On input public parameter* pp *this probabilistic algorithm outputs two binary stings, a secret key* sk *and a public key* pk*.*

$\mathsf{VRFEval}(sk, x)$**:** *On input a secret key* sk *and an input* $x \in \{0,1\}^{\ell(\lambda)}$*, this algorithm outputs* (v, π) *for the VRF value* $v \in \{0,1\}^{m(\lambda)}$ *and the corresponding proof* π *proving the correctness of* v*.*

$\mathsf{Verify}_{pk}(v, x, \pi)$**:** *On input* (pk, v, x, π)*, this probabilistic algorithm outputs either* 1 *or* 0*.*

A VRF is required to have the following security properties [39]:

Provability: *If* (v, π) *is the output of* $\mathsf{VRFEval}(sk, x)$*, then* $\mathsf{Verify}_{pk}(v, x, \pi)$ *outputs* 1*.*

Pseudorandomness: *Let* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *be a polynomial-time adversary playing the following experiment* Exp-PRand*:*

1. $pp \leftarrow \mathsf{ParamGen}(1^\lambda)$
2. $(pk, sk) \leftarrow \mathsf{KeyGen}(pp)$
3. $(x, st) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathsf{VRFEval}}(\cdot)}(pk)$
4. $(v_0, \pi_0) \leftarrow \mathsf{VRFEval}(sk, x)$
5. $v_1 \xleftarrow{\$} \{0,1\}^{m(\lambda)}$
6. $b \xleftarrow{\$} \{0,1\}$
7. $b' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{VRFEval}}(\cdot)}(v_b, st)$

where $\mathcal{O}_{\mathsf{VRFEval}}(\cdot)$ *is an oracle that on input a value* x *outputs the VRF value* v *and the corresponding proof of correctness* $\pi(sk, x)$*.*

The adversary $\mathcal{A}$ *that did not issue any queries to* $\mathcal{O}_{\mathsf{VRFEval}}$ *on the value* x*, wins the above game with probability:*

$$\Pr\left[b = b' \,|\, \mathcal{A} \text{ runs } \mathsf{Exp\text{-}PRand}\right] \leq \frac{1}{2} + \mathrm{negl}(\lambda).$$

Unconditional Full Uniqueness: *No values* $(pk, v_1, v_2, x, \pi_1, \pi_2)$ *can satisfy* $\mathsf{Verify}_{pk}(v_1, x, \pi_1) = \mathsf{Verify}_{pk}(v_2, x, \pi_2) = 1$ *when* $v_1 \neq v_2$*.*

In our work, we make two modifications to the above standard VRF security model. First, we use a k-time variant of the pseudorandomness property, where the $\mathcal{O}_{\mathsf{VRFEval}}(\cdot)$ oracle can be queried at most $k-1$ times by the

adversary (together with the challenge query to $\mathsf{VRFEval}(\cdot)$ in the pseudorandomness experiment, this gives a total of k $\mathsf{VRFEval}(\cdot)$ queries in the experiment). We also define the VRF output space to be $\bar{\mathcal{R}}_p$ (which is determined by our scheme's public parameters pp), rather than $\{0,1\}^{m(\lambda)}$ used in the original definition. The latter change does not introduce any difficulties since a pseudorandom output in $\bar{\mathcal{R}}_p$ can be easily mapped into a pseudorandom binary string with a cryptographic hash function or a randomness extractor.

Second, we slightly modify the "Unconditional Full Uniqueness" property of a VRF to a weaker "Computational Full Uniqueness", where the adversary is assumed to run in polynomial time. In particular, we define it as follows.

Definition 2.5 (Computational Full Uniqueness). *Let* $pp \leftarrow \mathsf{ParamGen}(1^\lambda)$. *A VRF is said to satisfy computational full uniqueness, if, on input pp, a polynomial-time adversary* $\mathcal{A}$ *outputs* $(x, pk, v_1, \pi_1, v_2, \pi_2) \leftarrow \mathcal{A}(pp)$ *such that* $\mathsf{Verify}_{pk}(v_1, x, \pi_1) = \mathsf{Verify}_{pk}(v_2, x, \pi_2) = 1$ *and* $v_1 \neq v_2$ *with at most* $\mathrm{negl}(\lambda)$ *probability.*

Remark 2.6. The notion of computational uniqueness has been first introduced in [24]. However, it is defined w.r.t. VRF without parameter generation algorithm $\mathsf{ParamGen}(1^\lambda)$, implying that public parameters can also be set maliciously. Such a notion was actually defined in the context of anonymous VRF, which is an extension of a standard VRF.

Remark 2.7. There is also a notion of computational *trusted* uniqueness [42] in the literature, in which one roughly requires that, given the VRF public key pk, each VRF input corresponds to a unique VRF output. The word "trusted" is basically used to indicate that the key generation process is trusted. Hence, in such a model, uniqueness with respect to untrusted key generation process is not a concern.

In the application of VRF to the blockchain consensus protocols [25,13], it was observed in [14] that pseudorandomness is not sufficient, and in fact an additional security property is needed, which is called the *unpredictability under malicious key generation* in [14, Section 3.2]. Informally, it means that an attacker that can maliciously choose the VRF key cannot bias the VRF output on a randomly chosen input, as long as the attacker has no information on the random input when choosing its VRF key. Accordingly, we formally define below an *unbiasability* property that captures this requirement in the same spirit with [14]. However, for consistency with the rest of our game-based security

notions, we provide a game-based definition, whereas the one in [14] is in the universal composability (UC) framework.

Unbiasability: Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a polynomial-time adversary playing the following experiment Exp-Bias:

1. $pp \leftarrow \mathsf{ParamGen}(1^\lambda)$
2. $(st, pk, v^*) \leftarrow \mathcal{A}_1(pp)$
3. $x \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}$
4. $(\pi, v) \leftarrow \mathcal{A}_2(x, st)$
5. $b \leftarrow \mathsf{Verify}_{pk}(v, x, \pi)$

$\mathcal{A}$ wins if $b = 1$ and $v = v^*$. We say that a VRF is *unbiasable* if

$$\Pr[\mathcal{A} \text{ wins Exp-Bias}] \leq 2^{-m(\lambda)} + \mathrm{negl}(\lambda).$$

3 Lattice-Based Few-Time Verifiable Random Function

3.1 k-time LB-VRF Construction

We use the parameter $k \in \mathbb{Z}^+$ to denote that a particular public-secret key pair output by KeyGen below is used to generate at most k VRF outputs. We further define the following challenge set:

$$\mathcal{C} = \{\, c \in \mathcal{R} \,:\, \|c\|_\infty \leq 1 \,\wedge\, \|c\|_1 \leq \kappa \,\}. \tag{1}$$

When performing operations over $\bar{\mathcal{R}}_p$, if a term $\mathbf{x}$ is initially defined over $\mathcal{R}$, then we first compute $\bar{\mathbf{x}} = \mathbf{x} \bmod (p, f(x))$ and then perform the remaining operations over $\bar{\mathcal{R}}_p$. For example, given $\mathbf{x} \in \mathcal{R}^s$ and $\mathbf{y} \in \bar{\mathcal{R}}_p^s$ for $s \geq 1$, $\langle \mathbf{x}, \mathbf{y} \rangle \in \bar{\mathcal{R}}_p$ indicates that $\langle \bar{\mathbf{x}}, \mathbf{y} \rangle$ is computed over $\bar{\mathcal{R}}_p$, where $\bar{\mathbf{x}} = \mathbf{x} \bmod (p, f(x))$.

$\mathsf{ParamGen}(1^\lambda)$**:** On input a security parameter λ, it outputs a public parameter $pp = (\mathbf{A}, G, H)$, where $G : \{0,1\}^* \to \bar{\mathcal{R}}_p^{n+\ell+k}$ and $H : \{0,1\}^* \to \mathcal{C}$ are two hash functions, and $\mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{n \times (n+\ell+k)}$.

$\mathsf{KeyGen}(pp)$**:** On input the public parameters pp, it randomly samples $\mathbf{s} \xleftarrow{\$} \mathbb{S}_1^{n+\ell+k}$, computes $\mathbf{t} = \mathbf{A} \cdot \mathbf{s} \in \mathcal{R}_q^n$ and outputs $pk = \mathbf{t}$ and $sk = \mathbf{s}$.

$\mathsf{VRFEval}(\mathbf{A}, \mathbf{t}, \mathbf{s}, \mu)$**:** On input public parameters pp, a public key $\mathbf{t}$, a secret key $\mathbf{s}$, and a message $\mu \in \{0,1\}^*$, perform the following.

1. Compute $\mathbf{b} = G(\mathbf{A}, \mathbf{t}, \mu) \in \bar{\mathcal{R}}_p^{n+\ell+k}$.
2. Compute $v = \langle \mathbf{b}, \mathbf{s} \rangle \in \bar{\mathcal{R}}_p$.
3. Pick $\mathbf{y} \xleftarrow{\$} \mathbb{S}_\beta^{n+\ell+k}$.
4. Compute $\mathbf{w}_1 = \mathbf{A} \cdot \mathbf{y} \in \mathcal{R}_q^n$.

5. Compute $w_2 = \langle \mathbf{b}, \mathbf{y} \rangle \in \bar{\mathcal{R}}_p$.
6. Compute $c = H(\mathbf{A}, \mathbf{t}, \mu, \mathbf{w}_1, w_2, v)$.
7. Compute $\mathbf{z} = \mathbf{y} + c \cdot \mathbf{s} \in \mathcal{R}^{n+\ell+k}$; if $\|\mathbf{z}\|_\infty > \beta - \kappa$ goto step 3.

The algorithm outputs the VRF proof $\pi := (\mathbf{z}, c)$ and the VRF value v.

$\mathsf{Verify}_{pk}(\pi, v, \mathbf{A}, \mu)$**:** On input VRF public key $pk = \mathbf{t}$, the VRF proof $\pi = (\mathbf{z}, c)$, the VRF value v, public parameter $\mathbf{A}$ and a message μ the algorithm computes:

1. Check $\|\mathbf{z}\|_\infty \stackrel{?}{\leq} \beta - \kappa$.
2. Compute $\mathbf{w}'_1 := \mathbf{A} \cdot \mathbf{z} - c \cdot \mathbf{t}$ over $\mathcal{R}_q$.
3. Compute $w'_2 := \langle \mathbf{b}, \mathbf{z} \rangle - c \cdot v$ over $\bar{\mathcal{R}}_p$ for $\mathbf{b} = G(\mathbf{A}, \mathbf{t}, \mu) \in \bar{\mathcal{R}}_p^{n+\ell+k}$.
4. Check $c \stackrel{?}{=} H(\mathbf{A}, \mathbf{t}, \mu, \mathbf{w}'_1, w'_2, v)$.

Table 3 summarizes the 3 different concrete parameter settings. For a detailed rationale behind these settings, please refer to Appendix A.

3.2 Security Analysis

The provability of our k-time LB-VRF construction follows via straightforward investigation. The pseudorandomness and unbiasability properties are also discussed and proved in Appendix B and C, respectively. We now focus on computational full uniqueness of our scheme.

Param.	**Explanation**	**Set I**	**Set II**	**Set III**
k	# of VRF outputs per key pair	1	3	5
d	$d = \dim(\mathcal{R}_q)$	256	256	256
q	prime $q \equiv 1 \bmod 2d$	100679681	$\sim 2^{26.8}$	$\sim 2^{27.1}$
p	prime $p \equiv 17 \bmod 32$	2097169	$\sim 2^{20}$	$\sim 2^{20}$
$\mathcal{R}_q$	polynomial ring $\mathbb{Z}_q[x]/(x^d + 1)$			
$f(x)$	a factor of $x^d + 1 \bmod p$	$x^{32} + 852368$		
$\bar{\mathcal{R}}_p$	polynomial ring $\mathbb{Z}_p[x]/(f(x))$			
n	MSIS rank	4	4	4
ℓ	MLWE rank	4	4	4
κ	Hamming weight of a challenge	39	39	39
β	max. coeff of masking randomness	89856	109824	129792
	average number of restarts	< 3	< 3	< 3
RHF	MSIS/MLWE root-Hermite factor	≈ 1.0045	≈ 1.0046	≈ 1.0047
Proof Size	size of a proof $(c, \mathbf{z})$	4.94 KB	6.13 KB	7.34 KB
VRF Size	size of a VRF evaluation v	84 Bytes	84 Bytes	84 Bytes
PK Size	size of a public key $\mathbf{t}$	3.32 KB	3.34 KB	3.39 KB

Table 3. Summary of identifiers and results of the parameter setting.

Computational Full Uniqueness. For the computational full uniqueness property, the following are the two main requirements:

- hardness of $\mathsf{MSIS}_{q,n,n+\ell+k,\gamma}$ for $\gamma = 8\kappa\beta\sqrt{n+\ell+k}$,
  - Looking ahead, this implies that (13) below holds without mod q and therefore also over $\bar{\mathcal{R}}_p$.
- any challenge difference is invertible in $\bar{\mathcal{R}}_p$.

Theorem 3.1 (Uniqueness). *Let $\gamma = 8\kappa\beta\sqrt{n+\ell+k}$ for the parameters $\kappa, \beta, n, \ell, k$ defined in Table 3 and assume that $\mathsf{MSIS}_{q,n,n+\ell+k,\gamma}$ is hard with $q > \gamma/2$. Further, let $p > 2^{20}$ be a prime such that $p \equiv 17 \bmod 32$. Then, k-time LB-VRF construction satisfies computational full uniqueness in the random oracle model.*

Proof. Let $\mathcal{A}$ be a PPT adversary against computational full uniqueness of k-time LB-VRF construction. We will show that two valid VRF evaluations produced by $\mathcal{A}$ on the same input must be the same, or else the $\mathsf{MSIS}_{q,n,n+\ell+k,\gamma}$ problem is solved, which occurs with negligible probability by the assumed hardness of the latter problem.

Let $\mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{n\times(n+\ell+k)}$, and G and H be two random oracles. Denote $pp = (\mathbf{A}, G, H)$ as the public parameters output by $\mathsf{ParamGen}$. Then, $\mathcal{A}(pp)$ outputs two valid VRF proof-evaluation pairs (π_0, v_0) with $\pi_0 = (\mathbf{z}_0, c_0)$ and (π_0', v_0') with $\pi_0' = (\mathbf{z}_0', c_0')$.

Rewind 1: Using a standard forking argument, we rewind $\mathcal{A}$ to the point $c_0 = H(\mathbf{A}, \mathbf{t}, \mu, \mathbf{A}\mathbf{z}_0 - c_0\mathbf{t}, \langle \mathbf{b}, \mathbf{z}_0\rangle - c_0 v_0, v_0)$ was queried, and return another challenge c_1 for the same input. With non-negligible probability, $\mathcal{A}$ produces another valid VRF output using c_1 such that $(\pi_1 = (\mathbf{z}_1, c_1), v_1)$ is a valid VRF proof-evaluation pair. Here, $\mathcal{A}$ may output a second valid pair, but we simply discard it.

Rewind 2: In a similar fashion as above, we rewind $\mathcal{A}$ to the point $c_0' = H(\mathbf{A}, \mathbf{t}, \mu, \mathbf{A}\mathbf{z}_0' - c_0'\mathbf{t}, \langle \mathbf{b}, \mathbf{z}_0'\rangle - c_0' v_0', v_0')$ was queried, and return another challenge c_1' for the same input. With non-negligible probability, $\mathcal{A}$ produces another valid VRF output using c_1' such that $(\pi_1' = (\mathbf{z}_1', c_1'), v_1')$ is a valid proof-evaluation pair. Again, $\mathcal{A}$ may output a second valid pair, but we simply discard it.

Overall, we have the following satisfied for $(\pi_0 = (\mathbf{z}_0, c_0), v_0), (\pi_1 = (\mathbf{z}_1, c_1), v_1), (\pi_0' = (\mathbf{z}_0', c_0'), v_0'), (\pi_1' = (\mathbf{z}_1', c_1'), v_1')$

$$\begin{aligned} &(\mathbf{A},\, \mathbf{t},\, \mu,\, \mathbf{A}\mathbf{z}_0 - c_0\mathbf{t},\, \langle \mathbf{b},\, \mathbf{z}_0\rangle - c_0 v_0,\, v_0) \\ =\,&(\mathbf{A},\, \mathbf{t},\, \mu,\, \mathbf{A}\mathbf{z}_1 - c_1\mathbf{t},\, \langle \mathbf{b},\, \mathbf{z}_1\rangle - c_1 v_1,\, v_1)\,, \end{aligned} \tag{2}$$

$$\begin{aligned} &(\mathbf{A},\, \mathbf{t},\, \mu,\, \mathbf{A}\mathbf{z}_0' - c_0'\mathbf{t},\, \langle \mathbf{b},\, \mathbf{z}_0'\rangle - c_0' v_0',\, v_0') \\ =\,&(\mathbf{A},\, \mathbf{t},\, \mu,\, \mathbf{A}\mathbf{z}_1' - c_1'\mathbf{t},\, \langle \mathbf{b},\, \mathbf{z}_1'\rangle - c_1' v_1',\, v_1')\,. \end{aligned} \tag{3}$$

The above two equalities implies the following

$$v_0 = v_1 =: v, \tag{4}$$
$$v_0' = v_1' =: v', \tag{5}$$
$$\mathbf{A}\mathbf{z}_0 - c_0\mathbf{t} = \mathbf{A}\mathbf{z}_1 - c_1\mathbf{t} \quad \text{over } \mathcal{R}_q, \tag{6}$$
$$\mathbf{A}\mathbf{z}_0' - c_0'\mathbf{t} = \mathbf{A}\mathbf{z}_1' - c_1'\mathbf{t} \quad \text{over } \mathcal{R}_q, \tag{7}$$
$$\langle \mathbf{b}, \mathbf{z}_0 \rangle - c_0 v_0 = \langle \mathbf{b}, \mathbf{z}_1 \rangle - c_1 v_1 \quad \text{over } \bar{\mathcal{R}}_p, \tag{8}$$
$$\langle \mathbf{b}, \mathbf{z}_0' \rangle - c_0' v_0' = \langle \mathbf{b}, \mathbf{z}_1' \rangle - c_1' v_1' \quad \text{over } \bar{\mathcal{R}}_p. \tag{9}$$

From now on, we stick to the notations v and v' due to (4) and (5). Rewriting (6) and (7), we get

$$\mathbf{A}(\mathbf{z}_0 - \mathbf{z}_1) = (c_0 - c_1)\mathbf{t}, \tag{10}$$
$$\mathbf{A}(\mathbf{z}_0' - \mathbf{z}_1') = (c_0' - c_1')\mathbf{t}, \tag{11}$$

Define $\bar{\mathbf{z}} := \mathbf{z}_0 - \mathbf{z}_1$, $\bar{\mathbf{z}}' := \mathbf{z}_0' - \mathbf{z}_1'$, $\bar{c} := c_0 - c_1$ and $\bar{c}' := c_0' - c_1'$. Multiplying (10) by $\bar{c}'$ and (11) by $\bar{c}$ and subtracting off the results, we get

$$\mathbf{A}\left(\bar{c}'\bar{\mathbf{z}} - \bar{c}\bar{\mathbf{z}}'\right) = \mathbf{0}. \tag{12}$$

Note that the following holds

$$\begin{aligned}\|\bar{c}'\bar{\mathbf{z}} - \bar{c}\bar{\mathbf{z}}'\| &\le \|\bar{c}'\bar{\mathbf{z}} - \bar{c}\bar{\mathbf{z}}'\|_\infty \cdot \sqrt{n+\ell+k} \le 2 \cdot \|\bar{c}'\|_1 \|\bar{\mathbf{z}}\|_\infty \cdot \sqrt{n+\ell+k} \\ &\le 2 \cdot 2\kappa \cdot 2\beta \cdot \sqrt{n+\ell+k} = 8\kappa\beta\sqrt{n+\ell+k}.\end{aligned}$$

By the assumption that $\mathsf{MSIS}_{q,n,n+\ell+k,\gamma}$ for $\gamma = 8\kappa\beta\sqrt{n+\ell+k}$ is hard, we conclude from (12) that, except for negligible probability,

$$\bar{c}'\bar{\mathbf{z}} = \bar{c}\bar{\mathbf{z}}' \quad \text{over } \mathcal{R}. \tag{13}$$

The fact that there is no mod q reduction comes from the following: $\|\bar{c}'\bar{\mathbf{z}}\|_\infty, \|\bar{c}\bar{\mathbf{z}}'\|_\infty < \gamma < q/2$.

Next, from (8), we get (replacing v_0 and v_1 with v)

$$\begin{aligned} & \langle \mathbf{b}, \mathbf{z}_0 \rangle - c_0 v = \langle \mathbf{b}, \mathbf{z}_1 \rangle - c_1 v \quad \text{over } \bar{\mathcal{R}}_p, \\ \iff \quad & \langle \mathbf{b}, \bar{\mathbf{z}} \rangle = \bar{c} v \quad \text{over } \bar{\mathcal{R}}_p. \end{aligned} \tag{14}$$

Similarly, from (9), we get

$$\langle \mathbf{b}, \bar{\mathbf{z}}' \rangle = \bar{c}' v' \quad \text{over } \bar{\mathcal{R}}_p. \tag{15}$$

Multiplying (14) by $\bar{c}'$ and (15) by $\bar{c}$, and then subtracting off the results, we get

$$\langle \mathbf{b}, \bar{c}'\bar{\mathbf{z}} - \bar{c}\bar{\mathbf{z}}' \rangle = \bar{c}\bar{c}' \left(v - v'\right) \quad \text{over } \bar{\mathcal{R}}_p. \tag{16}$$

Now since (13) holds over $\mathcal{R}$, by reducing mod p, it also holds over $\mathcal{R}_p$, and by further reducing mod f it also holds over $\bar{\mathcal{R}}_p$. Therefore, the left-hand side of (16) is equal to 0. By the assumption on p and Lemma 2.3, any challenge difference is invertible in $\mathcal{R}_p$ and thus also in $\bar{\mathcal{R}}_p$. This implies that $v = v'$. □

4 Implementation

4.1 Implementation of LB-VRF

We implemented Set I parameters (see Table 3) of our LB-VRF using Rust language. The source code of our implementation is available on GitHub[7]. The core operations are ring arithmetic over $\mathcal{R}_q$ and $\bar{\mathcal{R}}_p$, hash functions, and extendable output functions. We use SHA512 as our hash function, and ChaCha20 to extend hash digests into vector $\mathbf{b}$ and challenge c. For ring multiplications, we use index based method for polynomial multiplications involving secret keys or challenges (both are ternary polynomials); school book multiplication for $\bar{\mathcal{R}}_p$; and NTT multiplications for $\mathcal{R}_q$. We also hand-picked p, q and $f(x)$ for efficient mod reduction. We leave architecture-dependent optimizations, such as AVX2, to future work.

Our tests were conducted over a MacBookPro 2018, with an Intel(R) Core(TM) i7-8559U CPU @ 2.70GHz. The benchmark was conducted with Rust's benchmark tool known as *criterion*. The benchmark data is shown in Table 1. One may see that although the speed of LB-VRF is on par with classical VRFs, the size is significantly increased. This is unfortunately an inherit drawback from the current state of post-quantum cryptography.

4.2 Integration into Algorand Blockchain

Algorand's TPS model. To illustrate our benchmark results better, it is important to understand the bottleneck of the current Algorand protocol. Algorand's mainnet currently employs over 1000 nodes. A node is a whale holder of tokens and is likely to self-elect as a voter. Algorand allows for roughly 5.4 MB of "payload" transmission per round as a result of their efficient consensus protocol. To break up this data, 1000 nodes

[7] `https://github.com/zhenfeizhang/lb-vrf`

implies 1000 ECVRF proofs, which is 80 KB of data. It is straightforward to see that the majority of the data is reserved for transactions. If we assume that a transaction is 1KB on average, with an additional 64 bytes of data for authentications, then Algorand allows for 5K transactions per block, or, roughly 1K transaction per second (TPS). Therefore, we estimate the Algorand TPS throughput as follows:

$$\mathrm{TPS}_1 = \frac{\text{payload size} - (\text{total VRF cost}) \times \#\text{nodes}}{(\text{transaction size} + \text{signature size}) \times \text{blocktime}}.$$

It is also important to distinguish two notions

- Blocksize: a block is a set of data that is agreed by all the participants. Typically, it consists of transactions.
- Payload: the data that is transmitted through the network, during a block time. Typically, it consists of transactions and VRF data.

Note that the VRF data are not included in a block. This is because, during the voting, different committee member may have different views of the (subset of) voters, and there does not need to be a global view. Alternatively speaking, the committee does not need to agree on all votes, as long as each committee member has seen enough votes.

Our model. We envision that our k-time VRF may be deployed as follows. A user commits to a k-time VRF public key, pk_1, at round n via publishing a hash digest, along with a signature, of the public key. This allows her to use this VRF key at any round after $n + t$ (for a suitable parameter $t \geq 0$) via transmitting both the VRF public key and the VRF output to the other voters. Additionally, she similarly commits to $\lceil t/k \rceil - 1$ more public keys at rounds $n+1, \ldots, n+\lceil t/k \rceil - 1$ (i.e., $s := \lceil t/k \rceil$ public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_s$ are committed in total). This additional step is to make sure that the user can still participate in consensus right after the first public key pk_1 is consumed (without having to wait for t rounds).

Now at some round $n + t + i$ for $i \geq 0$, the VRF output is accepted if it is verified under the public key which hashes to the committed digest at round n. If the user has used k time of the VRF already, it needs to commit to a new public key and start using pk_2 in the next round(s). On the other hand, if the VRF output does not result into a winning ticket, the user does not publish anything, and thus, retains its ability to use the VRF key in future rounds. Overall, the user always has exactly s "usable" public keys committed on blockchain and uses the earliest committed key in creating the VRF output. The verifier therefore needs

to scan the blocks starting from n to make sure that the user is using the first of s VRF public keys committed. This step prevents the users from choosing between committed public keys.

We remark that for a given round n_1, the user may choose not to publish its public key even if it wins the lottery, and therefore retains its current public key for a later round n_2. This also happens for ECVRF where the user can forfeit a winning ticket. This does not harm the security as long as the user cannot predict the VRF output for n_2. Our VRF protocol requires the user to commit to a VRF public key at least t blocks prior to using it.

Since our VRF data becomes non-negligible compared to the whole payload, for a fair comparison, we set the network payload size as an invariant in our estimation, rather than the blocksize. In addition, we also alter the content in a block. For a k-time VRF, the user needs to commit to the VRF public key several rounds prior to using it. This commitment must be recorded in a block.

Concretely, in our model, we further set $k = 1$ to minimize VRF cost, and have the following estimation:

$$\mathrm{TPS}_2 = \frac{\text{payload size} - (\text{total VRF cost} + \text{digest} + \text{signature}) \times \#\text{nodes}}{(\text{transaction size} + \text{signature size}) \times \text{blocktime}}.$$

Our estimation. It is easy to see that our LB-VRF cannot scale to 1K nodes as 1K nodes already imply 8 MB of LB-VRF data (see below). We therefore compare our scheme with a maximum of 500 nodes. We note that although this number is smaller than the current status of Algorand, it is already sufficient for large blockchain platforms, and already exceeds the number of nodes of Algorand when it was launched.

Using the above formula, we computed estimates for the TPS of Algorand using our LB-VRF in combination with a variety of post-quantum signature schemes, in Table 2. In this computation, we make the following assumptions. We assume a payload size of 5.4 MB. Since our LB-VRF is a one-time VRF, we assume that in the Algorand consensus protocol, a node publishes both the VRF output, as well as the next VRF public key the node is committed to use. Therefore, in the TPS estimation formula above, we take the total VRF cost to be the sum of LB-VRF's VRF size (84 bytes), proof size (4.94 KB), and public-key length (3.32 KB), which is around 8 KB, using parameter set I in Table 3. In addition, we also require 32 bytes hash digest for the next VRF public key, and a signature of various bytes (see below) to authenticate that VRF public key. As done for Algorand, we assume 1 KB data for a transaction size. As Algorand

generates a block in about 5 seconds, we take blocktime as 5 seconds[8]. The last moving part in the equation is the signature size, which we set as 64 bytes for Ed25519, 700 bytes for Falcon[9], 2 KB for Dilithium [18], and 66 bytes for Rainbow[10].

Note that the post-quantum security of the signature scheme used in the consensus protocol is not of as a big concern as the VRF because the adversary cannot affect the consensus steps in the past by breaking the signature scheme. Therefore, until the quantum threat is imminent, one may opt to keep using Ed25519 as the signature in the hybrid "LB-VRF + Ed25519" mode, and switch to a post-quantum signature only when large-scale quantum computers are expected very soon.

Our result. We test our LB-VRF with a modified version of Algorand's reference implementation [2]. Our tests were conducted over a testnet with a maximum of 8 nodes, due to the limitation of resources. We use pingpong package from [2] to initiate transactions with increasing rates till the network saturates. We deploy the test over ARDC Nectar Research Cloud. Each node is a m3.small instance with 2 vCPU, 30GB Disk and 4GB RAM. The results are summarised in Table 4.

Setting	Metrics	Algorand with ECVRF	Algorand with LB-VRF
2 Nodes	TPS	771	746
	Tx per block	3322	3251
	Block time	4.30 second	4.35 second
4 Nodes	TPS	941	836
	Tx per block	4255	3648
	Block time	4.52 second	4.36 second
8 Nodes	TPS	821	811
	Tx per block	3637	3525
	Block time	4.42 second	4.34 second

Table 4. Cloud experiment with variable number of nodes

We remark that the main purpose of this test is not to show how TPS progresses with increasing number of nodes, which will be largely dominated by the network topology. Indeed, the current behaviour is inconclusive in that respect. Instead, we want to observe the impact of

[8] This is a rough estimation in September 2020. Since then, Algorand has upgraded its consensus for faster block generation time.

[9] `https://falcon-sign.info/`

[10] `https://www.pqcrainbow.org/`

replacing an ECVRF with an LB-VRF, for a same network setting. As expected, both the TPS and Tx per block is reduced slightly, for both local setting and cloud setting. The block generation time is slightly slower for the cloud setting due to the network latency. On the other hand, we only see marginal differences for block generation time between the ECVRF version and the LB-VRF version. For small networks, when the blockchain network cannot fully utilize the TPS, we argue that a LB-VRF based consensus is practical.

Acknowledgments. This work was supported in part by Australian Research Council Discovery Grant DP180102199 and also by use of the Nectar Research Cloud, a collaborative Australian research platform supported by the National Collaborative Research Infrastructure Strategy (NCRIS).

References

1. M. Abdalla, D. Catalano, and D. Fiore. Verifiable random functions from identity-based key encapsulation. In *Advances in Cryptology - EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 554–571. Springer, 2009.
2. Algorand. Goal: A reference implementation of Algorand. `https://github.com/algorand/go-algorand`.
3. Algorand. Reference implementation of BLS signature. `https://github.com/algorand/bls_sigs_ref`.
4. Algorand. Source code of ECVRF. `https://github.com/algorand/libsodium`.
5. A. Banerjee, C. Peikert, and A. Rosen. Pseudorandom functions and lattices. In *Proc. of EUROCRYPT*, volume 7237 of *LNCS*, pages 719–737. Springer, 2012.
6. E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *IEEE Symposium on Security and Privacy, S&P*, pages 459–474. IEEE Computer Society, 2014.
7. N. Bitansky. Verifiable random functions from non-interactive witness-indistinguishable proofs. In *Theory of Cryptography - 15th International Conference, TCC 2017*, volume 10678 of *LNCS*, pages 567–594. Springer, 2017.
8. D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *EUROCRYPT 2003*, 2003.
9. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. In *ASIACRYPT 2001*, 2001.
10. Z. Brakerski, S. Goldwasser, G. N. Rothblum, and V. Vaikuntanathan. Weak verifiable random functions. In *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC*, volume 5444 of *LNCS*, pages 558–576. Springer, 2009.
11. Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé. Classical hardness of learning with errors. In *STOC*, pages 575–584. ACM, 2013.
12. M. Chase and A. Lysyanskaya. Simulatable vrfs with applications to multi-theorem NIZK. In *Advances in Cryptology - CRYPTO 2007*, volume 4622 of *LNCS*, pages 303–322. Springer, 2007.
13. J. Chen, S. Gorbunov, S. Micali, and G. Vlachos. ALGORAND AGREEMENT: super fast and partition resilient byzantine agreement. *IACR Cryptol. ePrint Arch.*, 2018:377, 2018.

14. B. David, P. Gazi, A. Kiayias, and A. Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *EUROCRYPT (2)*, volume 10821 of *LNCS*, pages 66–98. Springer, 2018.
15. R. del Pino, V. Lyubashevsky, and G. Seiler. Lattice-based group signatures and zero-knowledge proofs of automorphism stability. In *ACM Conference on Computer and Communications Security*, pages 574–591. ACM, 2018.
16. Y. Dodis. Efficient construction of (distributed) verifiable random functions. In *Public Key Cryptography - PKC 2003*, volume 2567 of *LNCS*, pages 1–17. Springer, 2003.
17. Y. Dodis and A. Yampolskiy. A verifiable random function with short proofs and keys. In *Public Key Cryptography - PKC*, volume 3386 of *LNCS*, pages 416–431. Springer, 2005.
18. L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):238–268, 2018.
19. M. F. Esgin. *Practice-Oriented Techniques in Lattice-Based Cryptography*. PhD thesis, Monash University, 5 2020. `https://bridges.monash.edu/articles/Practice-Oriented_Techniques_in_Lattice-Based_Cryptography/12279728`.
20. M. F. Esgin, N. K. Nguyen, and G. Seiler. Practical exact proofs from lattices: New techniques to exploit fully-splitting rings. In *ASIACRYPT (2)*, volume 12492 of *LNCS*, pages 259–288. Springer, 2020.
21. M. F. Esgin, R. Steinfeld, J. K. Liu, and D. Liu. Lattice-based zero-knowledge proofs: New techniques for shorter and faster constructions and applications. In *CRYPTO (1)*, volume 11692 of *LNCS*, pages 115–146. Springer, 2019.
22. M. F. Esgin, R. K. Zhao, R. Steinfeld, J. K. Liu, and D. Liu. MatRiCT: Efficient, scalable and post-quantum blockchain confidential transactions protocol. In *ACM Conference on Computer and Communications Security*, pages 567–584. ACM, 2019. (Full version at `https://eprint.iacr.org/2019/1287`).
23. G. Fuchsbauer. Constrained verifiable random functions. In *Security and Cryptography for Networks - 9th International Conference, SCN 2014*, volume 8642 of *LNCS*, pages 95–114. Springer, 2014.
24. C. Ganesh, C. Orlandi, and D. Tschudi. Proof-of-stake protocols for privacy-aware blockchains. In *EUROCRYPT (1)*, volume 11476 of *LNCS*, pages 690–719. Springer, 2019.
25. Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*, pages 51–68. ACM, 2017.
26. S. Goldberg, M. Naor, D. Papadopoulos, L. Reyzin, S. Vasant, and A. Ziv. NSEC5: provably preventing DNSSEC zone enumeration. In *NDSS 2015*, 2015.
27. S. Goldberg, L. Reyzin, D. Papadopoulos, and J. Včelák. Verifiable random functions (vrfs). Internet Engineering Task Force, jun 2020. Full version available at `https://tools.ietf.org/html/draft-irtf-cfrg-vrf-07`.
28. O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions (extended abstract). In *25th Annual Symposium on Foundations of Computer Science, West Palm Beach, Florida, USA, 24-26 October 1984*, pages 464–479. IEEE Computer Society, 1984.
29. R. Goyal, S. Hohenberger, V. Koppula, and B. Waters. A generic approach to constructing and proving verifiable random functions. In *TCC (2)*, volume 10678 of *LNCS*, pages 537–566. Springer, 2017.

30. T. Hanke, M. Movahedi, and D. Williams. Dfinity technology overview series, consensus system. *arXiv preprint arXiv:1805.04548*, 2018.

31. S. Hohenberger and B. Waters. Constructing verifiable random functions with large input spaces. In *Advances in Cryptology - EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 656–672. Springer, 2010.

32. D. Hopwood, S. Bowe, T. Hornby, and N. Wilcox. Zcash protocol specification. Version 2020.1.14 [Overwinter+Sapling+Blossom+Heartwood+Canopy], 2020. `https://github.com/zcash/zips/blob/master/protocol/protocol.pdf`.

33. T. Jager. Verifiable random functions from weaker assumptions. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC*, volume 9015 of *LNCS*, pages 121–143. Springer, 2015.

34. A. Langlois and D. Stehlé. Worst-case to average-case reductions for module lattices. *Des. Codes Cryptogr.*, 75(3):565–599, 2015.

35. B. Libert, S. Ling, K. Nguyen, and H. Wang. Zero-knowledge arguments for lattice-based prfs and applications to e-cash. In *ASIACRYPT (3)*, volume 10626 of *LNCS*, pages 304–335. Springer, 2017.

36. A. Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In *CRYPTO*, volume 2442 of *LNCS*, pages 597–612. Springer, 2002.

37. V. Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In *Proc. of ASIACRYPT*, pages 598–616. Springer, 2009.

38. V. Lyubashevsky and G. Seiler. Short, invertible elements in partially splitting cyclotomic rings and applications to lattice-based zero-knowledge proofs. In *EUROCRYPT (1)*, volume 10820 of *LNCS*, pages 204–224. Springer, 2018.

39. S. Micali, M. O. Rabin, and S. P. Vadhan. Verifiable random functions. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 120–130. IEEE Computer Society, 1999.

40. I. Miers, C. Garman, M. Green, and A. D. Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *2013 IEEE SP 2013, 2013*, pages 397–411. IEEE Computer Society, 2013.

41. S. Noether and A. Mackenzie. Ring confidential transactions. *Ledger*, 1:1–18, 2016.

42. D. Papadopoulos, D. Wessels, S. Huque, M. Naor, J. Včelák, L. Reyzin, and S. Goldberg. Making nsec5 practical for dnssec. Cryptology ePrint Archive, Report 2017/099, 2017. `https://eprint.iacr.org/2017/099`.

43. S. Sun, M. H. Au, J. K. Liu, and T. H. Yuen. RingCT 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero. In *ESORICS (2)*, volume 10493 of *LNCS*, pages 456–474. Springer, 2017.

44. N. van Saberhagen. Cryptonote v 1.0, 2012. `https://cryptonote.org/whitepaper_v1.pdf`.

45. R. Yang, M. H. Au, Z. Zhang, Q. Xu, Z. Yu, and W. Whyte. Efficient lattice-based zero-knowledge arguments with standard soundness: Construction and applications. In *CRYPTO (1)*, volume 11692 of *LNCS*, pages 147–175. Springer, 2019.

46. T. H. Yuen, S. Sun, J. K. Liu, M. H. Au, M. F. Esgin, Q. Zhang, and D. Gu. RingCT 3.0 for blockchain confidential transaction: Shorter size and stronger security. In *Financial Cryptography*, volume 12059 of *LNCS*, pages 464–483. Springer, 2020.

A Concrete Parameter Setting Rationale

We first summarize the requirements on the parameters of our k-time VRF construction.

- $\beta \approx \kappa d(n+\ell+k)$,
- hardness of $\mathsf{MLWE}_{q,n+k,\ell,\chi}$ for $\chi = \mathcal{U}(\mathbb{S}_1)$,
- hardness of $\mathsf{MSIS}_{q,n,n+\ell+k,\gamma}$ for $\gamma = 8\kappa\beta\sqrt{n+\ell+k}$,
- invertibility of challenge differences in $\bar{\mathcal{R}}_p$.

We refer to Section 3.2 for more details and rigorous analysis.

Recall that our $\mathsf{VRFEval}$ uses rejection samplings to seal the information leakage on $\mathbf{z}$. In order to have a constant number of restarts on average, we simply set $\beta = \kappa d(n+\ell+k)$. Then, we fix $d = 256$ to balance the following aspects: (i) flexibility in implementation, (ii) the size of v and (iii) efficient polynomial arithmetic. Then, we set $\kappa = 39$ so that the challenge set $\mathcal{C}$ is sufficiently large, in particular, $|\mathcal{C}| > 2^{192}$. Then, we need to set (q, n, ℓ) to make sure that MSIS and MLWE are hard against best known attacks.

In order to estimate the hardness of MSIS and MLWE in practice against known attacks, we follow the methodology detailed in [19, Section 3.2.4]. In particular, we aim for a "root-Hermite factor" of $\delta \approx 1.0045$. The same target root-Hermite factor has been used in various works, e.g., [21,22] when aiming for 128-bit post-quantum security.

Finally, to argue invertibility of challenge differences in $\bar{\mathcal{R}}_p$, we rely on the results of [38] given in Lemma 2.3. By Lemma 2.3, it follows that if $p > 2^{20}$ and p is a prime with $p \equiv 17 \bmod 32$, then the difference of any two challenges in $\mathcal{C}$ is invertible in $\mathcal{R}_p$. Since $\bar{\mathcal{R}}_p$ as defined in Table 3 is one of the factors of $\mathcal{R}_p$, our parameter setting satisfies the challenge difference invertibility condition in $\bar{\mathcal{R}}_p$. For a summary of the parameters of our scheme and our parameter sets, we refer to Table 3. The first set of parameters represents the exact values used in our implementation while the last two do not specify q, p and $f(x)$ concretely.

It is clear that the main cost of the VRF proof size is due to the vector $\mathbf{z} \in \mathcal{R}^{n+\ell+k}$, which of $d(n+\ell+k)$ dimension if we see it as a vector over $\mathbb{Z}$. From MSIS and MLWE security, the values of nd and $n\ell$ are roughly fixed. However, the additional cost due to dk can be significantly smaller if d is decreased. Especially for $k > 1$, choosing d smaller than 256 results in smaller VRF proof lengths. But, we avoided this to have a fixed d for all parameter sets.

B Pseudorandomness

To prove pseudorandomness, we first assume that all the computations in $\mathsf{VRFEval}$ are done over $\mathcal{R}_q$ (i.e., $\bar{\mathcal{R}}_p = \mathcal{R}_q$). Then, we argue why changing some computations to be done over $\bar{\mathcal{R}}_p$ does not affect the pseudorandomness.

Assume that all the computations in $\mathsf{VRFEval}$ are done over $\mathcal{R}_q$. Let $pp = (\mathbf{A}, G, H) \leftarrow \mathsf{ParamGen}(1^\lambda)$ and suppose that k VRF outputs $(\pi_1, v_1), \ldots, (\pi_k, v_k)$ are generated for messages $\mu_1, \ldots, \mu_k$ under a public-secret key pair $(pk, sk) = (\mathbf{t}, \mathbf{s}) \leftarrow \mathsf{KeyGen}(pp)$, where $\pi_i = (c_i, \mathbf{z}_i)$. Then, we can write

$$\hat{\mathbf{t}} := \begin{pmatrix} \mathbf{t} \\ v_1 \\ \vdots \\ v_k \end{pmatrix} = \begin{pmatrix} \mathbf{A} \\ \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_k \end{pmatrix} \mathbf{s}, \tag{17}$$

where $\mathbf{b}_i = G(\mathbf{A}, \mathbf{t}, \mu_i)$ for $i = 1, \ldots, k$ ($\mathbf{b}_i$'s are treated as row vectors). The above structure is precisely a commitment to the zero vector over $\mathcal{R}_q$ for the "hashed-message commitment" (HMC) scheme in [22]. As discussed in [22, Appendix C], $\hat{\mathbf{t}}$ is computationally indistinguishable from a uniformly random element in $\mathcal{R}_q^{n+k}$ if $\mathsf{MLWE}_{q,n+\ell+k,\ell,\chi}$ for $\chi = \mathcal{U}(\mathbb{S}_1)$ is hard (see also [19, Lemma 3.4] for more details). This argument forms the basis for the following result.

Theorem B.1 (Pseudorandomness). *For the parameters q, n, k, ℓ and $\chi = \mathcal{U}(\mathbb{S}_1)$, assume that $\mathsf{MLWE}_{q,n+\ell+k,\ell,\chi}$ is hard and q is prime with $\frac{(n+\ell)d}{q^{\ell+1}}$ negligible in λ. Then, no PPT adversary can win $\mathsf{Exp\text{-}PRand}$ with non-negligible probability while making at most $k-1$ $\mathcal{O}_{\mathsf{VRFEval}}(\cdot)$ queries against the k-time LB-VRF construction in the random oracle model, when $\bar{\mathcal{R}}_p = \mathcal{R}_q$.*

Proof. We use the simulation of the zero-knowledge proof underlying our LB-VRF construction. Let $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{LWE}}$ be the advantage of $\mathcal{A}$ over solving $\mathsf{MLWE}_{q,n+\ell+k,\ell,\chi}$ for $\chi = \mathcal{U}(\mathbb{S}_1)$.

Game$_0$: This is identical to $\mathsf{Exp\text{-}PRand}$.

Game$_1$: First, the challenger simulates the response $\mathbf{z}$, where the rejection sampling is applied. That is, in $\mathsf{VRFEval}$, it replaces $\mathbf{z}$ by a uniformly random element in $\mathbb{S}_{\beta-\kappa}^{n+\ell+k}$. This game is perfectly indistinguishable from the previous game due to rejection sampling [37].

$$\mathrm{Adv}_{\mathcal{A}}^{\mathbf{Game}_0} - \mathrm{Adv}_{\mathcal{A}}^{\mathbf{Game}_1} = 0.$$

$\mathbf{Game}_2$: In $\mathsf{VRFEval}$, the challenger replaces v by a uniformly random element in $\bar{\mathcal{R}}_p = \mathcal{R}_q$. It follows from the discussion given before the theorem that this game is computationally indistinguishable from $\mathbf{Game}_1$ by $\mathsf{MLWE}_{q,n+\ell+k,\ell,\chi}$ assumption for $\chi = \mathcal{U}(\mathbb{S}_1)$. Note that in our $\mathsf{Exp\text{-}PRand}$ experiment, the adversary can make at most $k-1$ queries to $\mathcal{O}_{\mathsf{VRFEval}}(\cdot)$.

$$\left|\mathrm{Adv}_{\mathcal{A}}^{\mathbf{Game}_1} - \mathrm{Adv}_{\mathcal{A}}^{\mathbf{Game}_2}\right| \leq \mathrm{Adv}_{\mathcal{A}}^{\mathrm{LWE}}.$$

Now, in $\mathbf{Game}_2$, the output of $\mathsf{VRFEval}$ is independent of the secret key and the input message. Therefore, in $\mathbf{Game}_2$, the challenger can simulate all $\mathcal{O}_{\mathsf{VRFEval}}(\cdot)$ queries and the output v_0 is perfectly indistinguishable from v_1. □

Now, in the real construction, the only change to the above discussion is that the top part of (17) corresponding to $\mathbf{t}$ holds over $\mathcal{R}_q$, while the remaining bottom part corresponding to $(v_1, \ldots, v_k)$ holds over $\bar{\mathcal{R}}_p$. The technique of using two distinct moduli for a similar commitment scheme has already been employed, e.g., in [15] (see Section 2.5 and Lemma 6.2 in [15]). The only difference in [15] is that the commitment key constructed by $\mathbf{A}$ and $\mathbf{b}_i$'s is more structured, which does not affect the hardness assumption. As in [15], we consider the MLWE hardness with respect to the larger modulus (i.e., $q \geq p$). It is well known that as the modulus gets smaller, the MLWE problem gets harder against known attacks. Moreover, $v_i = \langle \mathbf{b}_i, \mathbf{s} \rangle$ over $\bar{\mathcal{R}}_p$ in our construction is not a full MLWE sample in $\mathcal{R}_p$, but rather only a *single* CRT coefficient of an MLWE sample in $\mathcal{R}_p$. The mapping $v_i = \langle \mathbf{b}_i, \mathbf{s} \rangle \mapsto v_i \bmod (f(x), p)$ maps the uniform distribution on $\mathcal{R}_p$ to the uniform distribution on $\bar{\mathcal{R}}_p$, so preserves the pseudorandomness of v_i (that's why the amount of information leaked in our construction is strictly less than the case of having $v_i = \langle \mathbf{b}_i, \mathbf{s} \rangle$ over $\mathcal{R}_p$). Therefore, we conclude that assuming the hardness of $\mathsf{MLWE}_{q,n+\ell+k,\ell,\chi}$ for $\chi = \mathcal{U}(\mathbb{S}_1)$, our LB-VRF construction satisfies pseudorandomness. For a more detailed discussion on the security reduction between two (M)LWE problems with distinct moduli, we refer the reader to [11] and [15, Section 6.1].

C Achieving Unbiasability

Extending our LB-VRF construction to be unbiasable is easy. In particular, we modify $\mathsf{VRFEval}$ such that it outputs $(\hat{v}, \pi)$, where $\pi = (\mathbf{z}, c, v)$ and $\hat{v} = \mathcal{H}(v, \mu)$ for some hash function $\mathcal{H}$ (modelled as a random oracle). In this case, the verification additionally needs to check that $\hat{v} = \mathcal{H}(v, \mu)$.

A similar idea was also used in [14]. Note that this extension does not affect the other properties (provability, pseudorandomness and uniqueness) as computation of $\hat{v} = \mathcal{H}(v, \mu)$ is a deterministic function with publicly known inputs.

As we work in the random oracle model, hashing a random input μ will result in a random output as long as the input has enough entropy. Therefore, for a given $\hat{v}^*$ and a random oracle $\mathcal{H}$,

$$\Pr_{\mu \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}} [\mathcal{H}(v, \mu) = \hat{v}^*] \leq \text{negl}(\lambda)$$

independent of how v is generated. From here, unbiasability of our k-time LB-VRF construction follows.