A Tale of Practical Verifiable Random Functions based on Post-Quantum Assumptions

Muhammed F. Esgin

Monash University
muhammed.esgin@monash.edu

Lattices meet Hashes Workshop, EPFL – May 2023

MONASH University

mfesgin.github.io

@mfesgin

Zero-knowledge

- ✓Formal
- ✓Reveal no info
- ✓Non-interactive
- ✓Minimal communication

This talk

- ✓Informal
- ✓As informative as possible
- ✓Interactive
- ✓Maximal communication

Outline

- What is a Verifiable Random Function (VRF)?

- Our post-quantum (PQ) VRF proposals
  - LB-VRF : first **practical** PQ VRF, from lattices (limited few-time pseudorandomness)
  - X-VRF : XMSS-based VRF (many-time but still limited and stateful)
  - SL-VRF : full-fledged VRF from symmetric primitives only
  - iVRF : an *indexed* VRF variant targeting blockchain apps
  - LaV : full-fledged VRF from lattices

- **Take away:** Use
  - iVRF for blockchain (e.g. Algorand-like systems)
  - LaV if you need full-fledged VRF

Comparison of properties

Scheme	Communication Size (bytes)	Key Homomorphism	Long Term	Stateless	Low Storage & Fast Keygen	Security Basis
SL-VRF ia.cr/2021/302	40,000*	✘	✓	✓	✓	Block cipher
LaV ia.cr/2022/141	12,000	✓	✓	✓	✓	Lattice
LB-VRF ia.cr/2020/1222	8,340**	✓	✘	✓	✓	Lattice
X-VRF ia.cr/2021/302	2,720	✘	Partial	✘	✘	Hash
iVRF ia.cr/2022/993	608	✘	Partial	✘	✘	Hash

*40KB size of SL-VRF is for LowMC block cipher. Using a more standard block cipher would likely further increase its size.

**LB-VRF comm. size includes the size of a public key since the construction is one-time, and therefore, requires continuous communication of PK.

Verifiable Random Function (VRF)

- Introduced by Micali, Rabin and Vadhan in 1999
- **Goal:** generate a secret-dependent random value in a verifiable fashion
- **Most prominent proposals:**
  - ECVRF based on elliptic curves - used by Algorand, Cardano
  - BLS-VRF based on pairings/BLS signature - used by Dfinity
- **Advantages of EC/BLS-VRF**
  - Does not require heavy machinery
  - Efficiency close to signature schemes
- **Applications**
  - Proof-of-Stake (PoS) based blockchain protocols
  - DNSSEC protocol
  - Key transparency: Google, Yahoo, and WhatsApp

NOT post-quantum secure!

Verifiable Random Functions – a bit more formally

- $pp \leftarrow \mathrm{ParamGen}(1^\lambda)$
- $(pk, sk) \leftarrow \mathrm{KeyGen}(pp)$
- $(v, \pi) \leftarrow \mathrm{VRFEval}_{sk}(x)$
- $0/1 \leftarrow \mathrm{Verify}_{pk}(v, x, \pi)$

Properties:

- **Provability**
  Verification of an honest VRF run succeeds
- **Pseudorandomness**
  VRF value v is pseudorandom (without π given)
- **Unconditional Full Uniqueness**
  For fixed (pk, x), there exists only a single VRF value v for which valid proof(s) can be created

Verifiable Random Functions – a bit more formally

- $pp \leftarrow \mathrm{ParamGen}(1^\lambda)$
- $(pk, sk) \leftarrow \mathrm{KeyGen}(pp)$
- $(v, \pi) \leftarrow \mathrm{VRFEval}_{sk}(x)$
- $0/1 \leftarrow \mathrm{Verify}_{pk}(v, x, \pi)$

Properties:

- **Provability**
  Verification of an honest VRF run succeeds
- ***κ*-Pseudorandomness**
  VRF value v is pseudorandom as long as adversary sees at most κ outputs
- **Unconditional Full Uniqueness**
  For fixed (pk, x), there exists only a single VRF value v for which valid proof(s) can be created

Verifiable Random Functions – a bit more formally

- $pp \leftarrow \text{ParamGen}(1^\lambda)$
- $(pk, sk) \leftarrow \text{KeyGen}(pp)$
- $(v, \pi) \leftarrow \text{VRFEval}_{sk}(x)$
- $0/1 \leftarrow \text{Verify}_{pk}(v, x, \pi)$

Properties:

- **Provability**
  Verification of an honest VRF run succeeds
- **$\boldsymbol{\kappa}$-Pseudorandomness**
  VRF value v is pseudorandom as long as adversary sees at most κ outputs
- **Computational Full Uniqueness**
  For fixed (pk, x), PPT adversary cannot create two distinct VRF values $v_1 \neq v_2$ along with valid proofs

A folklore VRF approach (using random oracles)

- Take a PRF (with certain properties)
- Glue it with a Non-Interactive Zero-Knowledge (NIZK) proof
  - to prove honest PRF evaluation in zero-knowledge
- That gives **verifiable PRF** => VRF

Desired PRF Properties

- **Notation**
  K: key space
  K': extended key space (to accommodate for relaxed NIZK soundness)
  T: output space
  R: underlying (commutative) scalar ring

- **Key-binding:**
  $$\Pr\left[(m, k_0, k_1) \leftarrow A \;:\; k_1 \neq k_0 \text{ and } \mathrm{PRF}_{k_1}(m) = \mathrm{PRF}_{k_0}(m)\right] < \mathrm{negl}$$
  - Can be *statistical* or *computational*

- **Additive key-homomorphism:**
  $$\mathrm{PRF}_{\alpha \cdot k_0 + k_1}(m) = \alpha \otimes \mathrm{PRF}_{k_0}(m) \oplus \mathrm{PRF}_{k_1}(m)$$
  for some homomorphism space $S \subseteq R$

NIZK

- We require NIZK to prove

$$\mathrm{Rel}_{\mathrm{vrf}} = \{(m, pk, v), (f, k) : f \otimes pk = \mathrm{PRF}_k(0), f \otimes v = \mathrm{PRF}_k(m),$$
$$f \in F \quad \text{and} \quad f' \cdot k \in K' \text{ for all } f' \in F\}$$

for a set $F \subseteq R$ of "relaxation factors"

- **Note:** $F = \{1\}$ for typical DL-based proofs
  - So, $K' = K$ is sufficient

Folklore VRF from PRF+NIZK

- **ParamGen:** generate NIZK public params and publish them

- **KeyGen:** Sample random $k \leftarrow K$, set $pk = \mathrm{PRF}_k(0)$ and $sk = k$

- **VRFEval(m, k):**
  - Compute $v = \mathrm{PRF}_k(m)$
  - Generate NIZK π for $\mathrm{Rel}_{\mathrm{vrf}}$
  - Output v as VRF value and π as VRF proof

$$\boxed{\begin{aligned}\mathrm{Rel}_{\mathrm{vrf}} = \{(m, pk, v), (f, k) : f \otimes pk = \mathrm{PRF}_k(0), f \otimes v = \mathrm{PRF}_k(m),\\ f \in F \quad \text{and} \quad f' \cdot k \in K' \text{ for all } f' \in F\}\end{aligned}}$$

- **Verify:** run NIZK verification

VRF Security discussion (informal)

- **Provability:** easy

- **$\boldsymbol{\kappa}$-Pseudorandomness:** follows from
  - NIZK simulatability
  - PRF κ-pseudorandomness

- **Uniqueness:** let's look more into it!

Uniqueness of Folklore VRF (with relaxed NIZK)

$$\mathrm{Rel_{vrf}} = \{(m, pk, v), (f, k) : f \otimes pk = \mathrm{PRF}_k(0), f \otimes v = \mathrm{PRF}_k(m), f \in F \ \text{ and } \ f' \cdot k \in K' \text{ for all } f' \in F\}$$

- Suppose (v_1, π_1) and (v_2, π_2) accepting for $(\mathrm{m}, \mathrm{pk})$
- Use NIZK extractor to get

$$f_1^* \otimes \mathsf{pk} = \mathsf{PRF}_{k_1^*}(0) \implies f_2^* \otimes f_1^* \otimes \mathsf{pk} = \mathsf{PRF}_{f_2^* \cdot k_1^*}(0), \tag{7}$$

$$f_1^* \otimes v_1 = \mathsf{PRF}_{k_1^*}(\mathsf{m}), \tag{8}$$

$$f_2^* \otimes \mathsf{pk} = \mathsf{PRF}_{k_2^*}(0) \implies f_1^* \otimes f_2^* \otimes \mathsf{pk} = \mathsf{PRF}_{f_1^* \cdot k_2^*}(0), \tag{9}$$

$$f_2^* \otimes v_2 = \mathsf{PRF}_{k_2^*}(\mathsf{m}). \tag{10}$$

No key-binding required!

- By PRF key-binding, (and commutativity of R)

$$f_2^* \cdot k_1^* = f_1^* \cdot k_2^* \qquad \text{over } R$$

- By PRF key-homomorphism,

$$f_2^* \otimes f_1^* \otimes v_1 = \mathsf{PRF}_{f_2^* \cdot k_1^*}(\mathsf{m})$$

- Assuming relaxation factors are invertible, we get

$$v_1 = v_2$$

Some instantiations of folklore VRF approach

ECVRF [PWH+17] (ia.cr/2017/099) (Discrete log based)

- DL-based PRF
  $$g^{s} = v$$
  where $g \leftarrow G(m)$ for random oracle G, and
  $k = s$ is a standard DL secret

- Easy to see homomorphism and statistical key-binding

- Glue it with DL proof of equality

LB-VRF [EKS+, FC'21] (Module-SIS and Module-LWE)

- MLWE-based PRF

$$\mathrm{PRF}_k(m) = \boldsymbol{A} \cdot \boldsymbol{s} + \boldsymbol{e}$$

  where $\boldsymbol{A} \leftarrow G(m)$ for random oracle G, and
  $k = \boldsymbol{s}$ is secret vector generated at KeyGen

- But, how about the error $\boldsymbol{e}$?
  - Can't let $\boldsymbol{e}$ chosen randomly each time in Eval: breaks uniqueness!
  - **Solution**: Fix $\boldsymbol{e}$ at KeyGen (and prove its use in Eval)

- Then, for each VRFEval, we reveal $(\boldsymbol{v_i}, \boldsymbol{A_i})$ for **fixed** $(\boldsymbol{s}, \boldsymbol{e})$
- So, can only Eval a few times (pk size $\sim O(\kappa)$)

MLWE-based PRF: Properties

- **Key-homomorphism:**
  Easy to see
  $\alpha \cdot (\boldsymbol{A} \cdot \boldsymbol{s}_1 + \boldsymbol{e}_1) + (\boldsymbol{A} \cdot \boldsymbol{s}_2 + \boldsymbol{e}_2) = \boldsymbol{A} \cdot (\alpha \cdot \boldsymbol{s}_1 + \boldsymbol{s}_2) + (\alpha \cdot \boldsymbol{e}_1 + \boldsymbol{e}_2)$

- **Key-binding:** Assume we find (relatively) short $(\boldsymbol{s}_1, \boldsymbol{e}_1) \neq (\boldsymbol{s}_2, \boldsymbol{e}_2)$

$$\boldsymbol{A} \cdot \boldsymbol{s}_1 + \boldsymbol{e}_1 = \boldsymbol{A} \cdot \boldsymbol{s}_2 + \boldsymbol{e}_2$$
$$\Rightarrow \quad [\, \boldsymbol{A} \,||\, \boldsymbol{I} \,] \cdot \begin{bmatrix} \boldsymbol{s}_1 - \boldsymbol{s}_2 \\ \boldsymbol{e}_1 - \boldsymbol{e}_2 \end{bmatrix} = \boldsymbol{0}$$

  Computationally hard due to Module-SIS!

- We show: relaxed NIZK proof of knowledge of short secret vector sufficient

LaV [ESLR22] (Module-SIS and Module-LWR)

- MLWR-based PRF

$$\mathrm{PRF}_k(m) = \lfloor \boldsymbol{A} \cdot \boldsymbol{s} \rceil_p$$

  where $\boldsymbol{A} \leftarrow G(m)$ for random oracle G, and
  $k = \boldsymbol{s}$ is secret vector generated at KeyGen (and p divides q)

- Now, have deterministic error
  - Can be generated at each Eval
  - Can evaluate for 2^{128} times (→ ever-lasting LOVE/LaV)

- Ok, but why not do this in the first place?
  - More difficult to prove rounding relation (in terms of efficiency)!

MLWR-based PRF: Properties

- **(Almost) Key homomorphism:**

$$\lfloor \boldsymbol{A} \cdot \boldsymbol{s_1} \rceil_p + \lfloor \boldsymbol{A} \cdot \boldsymbol{s_2} \rceil_p \approx \lfloor \boldsymbol{A} \cdot (\boldsymbol{s_1} + \boldsymbol{s_2}) \rceil_p$$

- **Key-binding:** Similarly argued as in MLWE-based PRF

How to prove rounding?

Fact

$\boldsymbol{v} = \lfloor \boldsymbol{u} \rceil_p$ for $p \mid q$ $\Leftrightarrow$

$\exists \boldsymbol{e} \in [q/p]^n$ with $\boldsymbol{e} = \boldsymbol{u} - \frac{q}{p} \cdot \boldsymbol{v} \bmod q$

1) Must really prove this exactly!

2) Try to exploit relaxed proofs

LANES[+]: Framework for Hybrid Exact/Relaxed Proofs

- Want to prove

$$\boldsymbol{Ar} + \boldsymbol{Bm} = \boldsymbol{t} \quad \text{over} \quad R_{q,d} = \mathbb{Z}_q[X]/(X^d + 1)$$

- Can use an exact proof
  - But less efficient than relaxed ones
  - Particularly gets costly with increasing witness dimension
  - Imagine want to prove knowledge of a single LWE sample: $\langle \boldsymbol{a}, \boldsymbol{s} \rangle + e$

- Interested in: exactness only needed for $\boldsymbol{m}$, but not for $\boldsymbol{r}$

- **Can we build an efficient hybrid exact/relaxed proof framework?**

LANES$^+$: Framework for Hybrid Exact/Relaxed Proofs

$$\mathcal{L}^+(\mathsf{mp},\mathsf{ulp}) = \left\{ (\bar{c},\mathbf{m},\mathbf{r},\vec{v}) : \begin{array}{c} \mathbf{t} = \mathbf{A}\mathbf{r} + \mathbf{B}\mathbf{m} \text{ over } \mathcal{R}_{q,d} \wedge \mathbf{G}_1\vec{\mathbf{m}} = \mathbf{G}_2\vec{v} \bmod q \\ \wedge P(\vec{\mathbf{m}},\vec{v}) = 0 \bmod q \ \forall P \in \mathsf{mp} \wedge \\ \|\bar{c}\mathbf{r}\|_\infty \le \gamma_r \wedge \|\bar{c}\|_\infty \le \gamma_c \text{ for } \gamma_r,\gamma_c \ll q \in \mathbb{Z}^+ \end{array} \right\}$$

where mp is a set of multivariate polynomials in the coordinates of $(\vec{\mathbf{m}},\vec{v})$ over $\mathbb{Z}_q$ (for example, enforcing the smallness of the witness coefficients via $P_i(\vec{\mathbf{m}},\vec{v}) = v_i(v_i-1)$ for $\vec{v} = (v_0, v_1, \ldots)$), $\mathsf{ulp} = ((\mathbf{A},\mathbf{B},\mathbf{t}),(\mathbf{G}_1,\mathbf{G}_2))$ is the collection of linear relations and γ_r,γ_c are some public norm-bounds.

- Need two ingredients: RPoK and LANES (or an exact proof)

Standard RPoK Proof

Algorithm 1 Standard Lattice-based Relaxed Proof of Knowledge (RPoK)

procedure RPoK$((\mathbf{A}, \mathbf{B}, \mathbf{t}); (\mathbf{r}, \mathbf{m}))$:
- Sample short rand. masking $\mathbf{y}$
- Sample message masking $\mathbf{u}$
- $\mathbf{w} = \mathbf{Ay} + \mathbf{Bu}$ over $\mathcal{R}_{q,d}$
- $c \leftarrow \mathcal{H}(\mathbf{A}, \mathbf{B}, \mathbf{t}, \mathbf{w})$ for a hash $\mathcal{H}$
- $\mathbf{z} = \mathbf{y} + c \cdot \mathbf{r}$
- $\mathbf{f} = \mathbf{u} + c \cdot \mathbf{m}$
- Rejection samp. on $\mathbf{z}$ (and $\mathbf{f}$ if req.)
- **return** proof $\pi = (c, \mathbf{z}, \mathbf{f})$

end procedure

procedure Verify$((\mathbf{A}, \mathbf{B}, \mathbf{t}), \pi)$:
- Parse $\pi = (c, \mathbf{z}, \mathbf{f})$
- If $\mathbf{z}$ (and $\mathbf{f}$) is not sufficiently short, **return** 0
- $\mathbf{w}' = \mathbf{Az} + \mathbf{Bf} - c\mathbf{t}$ over $\mathcal{R}_{q,d}$
- If $c \neq \mathcal{H}(\mathbf{A}, \mathbf{B}, \mathbf{t}, \mathbf{w}')$, **return** 0
- **return** 1

end procedure

- Proves: $\bar{c} \cdot \boldsymbol{t} = \boldsymbol{Am}' + \boldsymbol{Br}'$ where $\boldsymbol{m}', \boldsymbol{r}', \bar{c}$ are (relatively) short

LANES Proof

- Exact (lattice) proof system due to a series of works
  - [ALS20] (ia.cr/2020/517)
  - [ENS20] (ia.cr/2020/518)
  - [LNS20] (ia.cr/2020/1183)

- Can prove linear and multiplicative relations:

$$\mathcal{L}(\mathsf{mp}, \mathsf{ulp}) = \left\{ \overrightarrow{m} \in \mathbb{Z}_q^{Nl} \; : \; \forall P \in \mathsf{mp}, \; P(\overrightarrow{m}) = \overrightarrow{0} \mod q \wedge \mathbf{A}\overrightarrow{m} = \overrightarrow{u} \mod q \right\}$$

LANES$^+$: RPoK + LANES

procedure $\mathsf{LANES}^+.\mathsf{Prove}_{\mathsf{pp}}((\mathsf{mp}, \mathsf{ulp}), (t; t'); \rho)$ $\triangleright$ ρ is optional; only used as $\mathcal{H}$ input

Parse $(t; t') = (t_L; (t'_L, \mathbf{m}, \mathbf{r}, \overrightarrow{v}, \mathbf{u}))$

Sample short randomness masking $\mathbf{y} \stackrel{\$}{\leftarrow} \mathbb{D}^{\dim(\mathbf{r})}_{\phi\eta, d}$

Compute $\mathbf{w} = \mathbf{A}\mathbf{y} + \mathbf{B}\mathbf{u}$

$c \leftarrow \mathcal{H}(\mathsf{pp}, \mathsf{mp}, \mathsf{ulp}, t, \mathbf{w}; \rho)$

$\mathbf{z} = \mathbf{y} + c \cdot \mathbf{r}$

$\mathbf{f} = \mathbf{u} + c \cdot \mathbf{m} \in \mathcal{R}^V_{q,d}$

Restart if $\mathsf{Rej}(\mathbf{z}, c\mathbf{r}, \phi, \eta)$

Restart if $\mathsf{flag}_{\mathsf{rs}} = \mathsf{true}$ and $\mathsf{Rej}(\mathbf{f}, c\mathbf{m}, \phi_m, \eta_m)$

$\mathsf{ulp}' = \left(\mathbf{L}, \begin{pmatrix} \overrightarrow{\mathbf{f}} \\ \overrightarrow{0} \end{pmatrix}\right)$ where $\mathbf{L} := \begin{pmatrix} \mathbf{I}_{Vd} & \mathbf{I}_V \otimes \mathsf{Rot}(c) & \mathbf{0} \\ \mathbf{0} & \mathbf{G}_1 & -\mathbf{G}_2 \end{pmatrix}$ $\dashrightarrow$ Multiplied by $\begin{pmatrix} \vec{\mathbf{u}} \\ \vec{\mathbf{m}} \\ \vec{v} \end{pmatrix}$

$\pi_L \leftarrow \mathsf{LANES}.\mathsf{Prove}_{\mathsf{pp}_L}((\mathsf{mp}, \mathsf{ulp}'), (t_L; t'_L))$

return the proof $\pi = (\pi_L, \hat{\pi})$ with $\hat{\pi} = (c, \mathbf{z}, \mathbf{f})$

end procedure

LANES^{+} (cont'd)

- Can support many other exact proofs
  - For example, LNP22 proof system
  - For small-dim $\boldsymbol{m}$, use LANES
  - For medium-dim $\boldsymbol{m}$, use LNP22

- Can support different moduli for RPoK and LANES
  - Important for VRF application (to use the rounding fact)

- No assumption on the shape of $\boldsymbol{B}$
  - Can be expanding

[LNP22] V. Lyubashevsky, N. K. Nguyen, and M. Plançon. Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. In CRYPTO, 2022

Back to LaV

- Instantiate LANES$^+$ to prove
  - $\frac{q}{p} \cdot \boldsymbol{v} = \boldsymbol{B}\boldsymbol{s} - \boldsymbol{e} \mod q,$ and
  - $\boldsymbol{e} \in [q/p]^n$ (using multiplicative proof in LANES)
- Shrink the dimension of $\boldsymbol{e}$ as much as possible
  - Concrete instantiation: e is a single polynomial of degree <32 with coefficients in $[0, \ldots, 61)$
  - i.e., only about $6 \cdot 32 = 192$ bits of entropy
  - However, entropy$(\boldsymbol{s}, \boldsymbol{e}) > 7{,}000$ bits typically
- **Costs**
  - Exact proof (LANES): 8.8 KB (can we do exact range proof more efficiently?)
  - Relaxed proof: 3.2 KB

Symmetric-key based VRFs

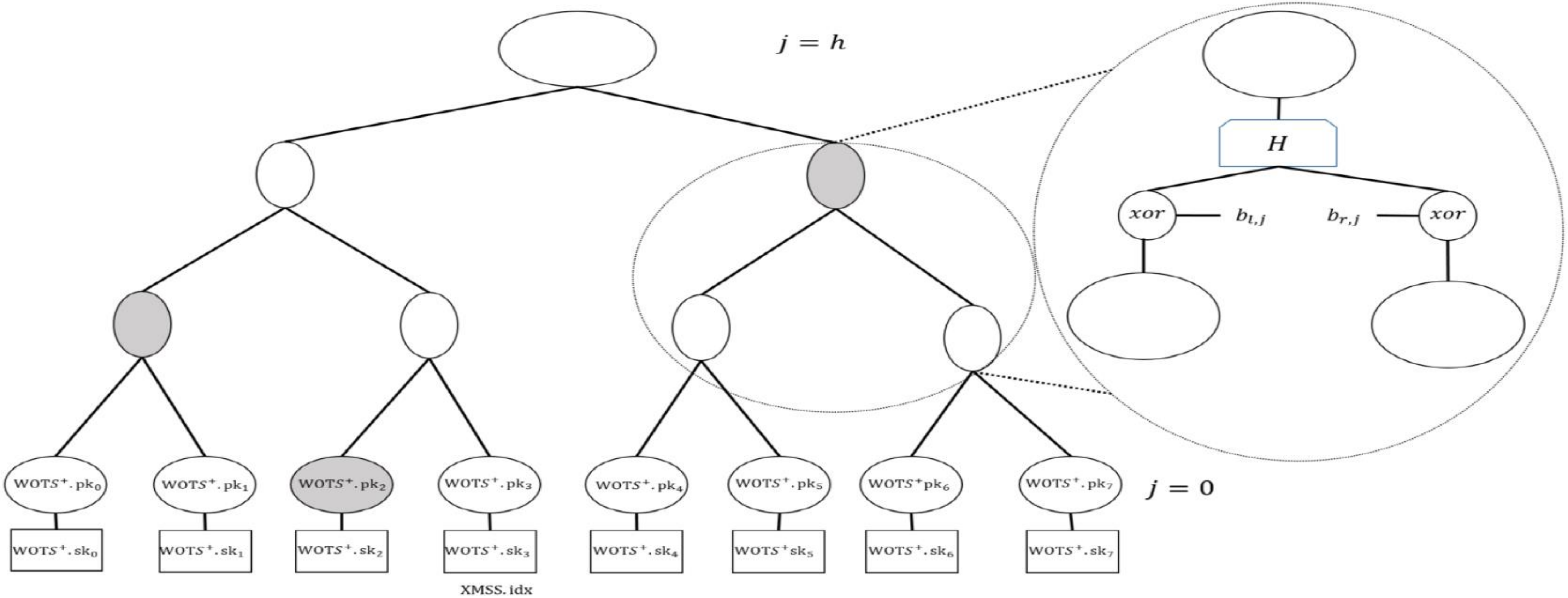
SL-VRF [BDE+, Esorics'22] (symmetric primitives)

- Take a symmetric-key based PRF
  - LowMC is used in the paper
- Glue it with symmetric-key based NIZK
  - KKW18 proof is used in the paper
- Used as a baseline

[KKW18] J. Katz, V. Kolesnikov, and X. Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In ACM CCS, 2018

X-VRF [BDE+, Esorics'22] (XMSS based)

- XMSS works as follows (briefly)



- XMSS sig = ($WOTS^+$ sig, idx, auth. path)

X-VRF (cont'd)

- Set $v = H(\mathrm{XMSS.sig}, m)$ for random oracle H

- WOTS$^+$ sig is an (iterated) hash of some random strings
- So pseudorandomness is easy to argue

- Uniqueness: a bit tricky
  - If signed w.r.t. different leaves, then uniqueness breaks down
  - Get around: Force users to sign w.r.t. a fixed leaf

indexed VRF (iVRF)

For blockchain

How common leader election approaches work

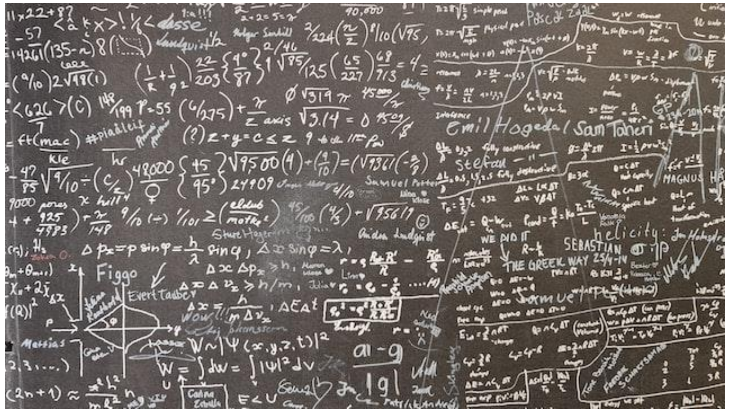
- **Goal:** Choose someone to decide on the next block

- n: the round number
- i: the user number

- Assume a random "magic" number Q_n at each round

A view of Algorand's approach

- Ask users to pick a private function H_{sk_i} **in advance**
  - Corresponding public key pk_i on chain ensures no change of H_{sk_i}
- At round n, users check if $v_{i,n} \coloneqq H_{sk_i}(Q_n) < T_{i,n}$ for a (stake-dependent) threshold $T_{i,n}$
- If successful, output $v_{i,n}$ and a crypt. proof $\pi_{i,n}$ that $v_{i,n} = H_{sk_i}(Q_n)$



A view of Bitcoin's approach

- At round n, use Q_n to randomly select a global one-way function H_{Q_n}
- Let users race real-time to find a "lucky" input x with $H_{Q_n}(x) < T$ for some threshold T

A view of our approach

- Ask users to (vector) commit to input $x_{i,n}$ **in advance**

- At round n, use Q_n to select a global random function H_{Q_n} (as Bitcoin)

- At round n, users check if $v_{i,n} := H_{Q_n}(x_{i,n}) < T_{i,n}$

- If successful, reveal $v_{i,n}$ and $x_{i,n}$

“Dual” view of our approach

- Ask users to (vector) commit to $x_{i,n}$ defining $H_{x_{i,n}}$ **in advance**
- At round n, use Q_n as a fixed global input
- At round n, users check if $v_{i,n} \coloneqq H_{x_{i,n}}(Q_n) < T_{i,n}$
- If successful, reveal $v_{i,n}$ and $H_{x_{i,n}}$ (i.e., $x_{i,n}$)
- No need for a (complicated) zero-knowledge proof

Ok, but what is this tool that we are using here?

- Want **uniqueness**: a user can generate a single $v_{i,n}$ at round n
- Want **pseudorandomness**: $v_{i,n}$ looks random

- Ok, so this is a VRF?

  Not quite!

- Seems like the (regular) VRF properties may not be the right fit for blockchain leader election

- Do **NOT** need pseudorandomness for past rounds!

Indexed VRF [EEK+, AsiaCCS 2023]

- VRF input additionally has an index (round number in blockchain)
- **Pseudorandomness:** only holds against "future" indices
- **Uniqueness:** only holds for a fixed (index, msg) pair
- This model seems to fit the blockchain setting better

Advantages of our iVRF approach

- **Simplicity and flexibility:** well-known, simple tools.
  Any signature and (cryptographic) hash can be used

- **Sustainability:** no racing condition ⇒ no need to compete for more power

- **Efficiency:** the extra cost for VRF functionality (on top of forward-secure signature) is just 32 bytes

- **(Post-Quantum) Security:** leader election part only uses hash (safest PQ option)
  - Security proof in the standard model
    No random oracles ⇒ no need for Quantum Random Oracle Model analysis

Performance Results

Efficiency comparison

Scheme	Proof Size (bytes)	Public Key Size (bytes)	VRF Value Size (bytes)	Keygen Time (ms)	Evaluation Time (ms)	Verification Time (ms)	Number of Evaluations
SL-VRF ia.cr/2021/302	40,000	48	32	0.38	765	475	Unlimited
LaV ia.cr/2022/141	12,000	6,400	124	-	-	-	Unlimited
LB-VRF ia.cr/2020/1222	5,000	3,300	84	0.33	3.10	1.30	1
X-VRF ia.cr/2021/302	2,720	64	32	426000	0.74	0.90	2^{18}
iVRF ia.cr/2022/993	608	32	0	< 3087	0.01	0.02	2^{18}
(non-PQ) EC-VRF ia.cr/2017/099	80	32	32	0.05	0.10	0.10	Unlimited

Open Questions

- Security analysis in Quantum Random Oracle Model (QROM)
  - All VRFs discussed are in ROM
  - Promising direction by Peikert and Xu for ECVRF (ia.cr/2023/223)
- More advanced VRF constructions like oblivious VRF
- More applications of LANES^{+} (work in progress)
- Efficient PQ VRF based on other assumptions
  - Recent isogeny-based work by Yi-Fu Lai: 35-40 KB proofs (ia.cr/2023/182)

Designated-Verifier zk-SNARKs from Lattices

	Base encryption	Quasi-optimal	ZK technique
ISW21 (ia.cr/2021/977)	MLWE-Regev	NO	Exponential smudging
Our work (ia.cr/2022/1690)	MLWE-HalfGSW	YES	Polynomial re-randomization

sec. level≈128 bits
$N=2^{20}$ constraints

	Proof Size (KB)	Compressed CRS Size (GB)	CRS Size (GB)
ISW21	33.0	10	337
Our work – VI	10.7	12	391
Our work – V	8.7	15	344
Our work – IV	6.5	26	410
Our work – III	6.1	36	454
Our work – II	5.8	76	442
Our work – I	5.3	252	1368

THANK YOU!

Full versions of these works and implementation codes:
mfesgin.github.io/publications/

Efficiency comparison

Scheme	Proof Size (bytes)	Public Key Size (bytes)	VRF Value Size (bytes)	Keygen Time (ms)	Evaluation Time (ms)	Verification Time (ms)	Number of Evaluations
SL-VRF ia.cr/2021/302	40,000	48	32	0.38	765	475	Unlimited
LaV ia.cr/2022/141	12,000	6400	124	-	-	-	Unlimited
LB-VRF ia.cr/2020/1222	5,000	3300	84	0.33	3.10	1.30	1
X-VRF ia.cr/2021/302	2,720	64	32	426000	0.74	0.90	2^{18}
iVRF ia.cr/2022/993	608	32	0	< 3087	0.01	0.02	2^{18}
(non-PQ) EC-VRF ia.cr/2017/099	80	32	32	0.05	0.10	0.10	Unlimited

34

Please feel free to contact me:
muhammed.esgin@monash.edu

MONASH University

mfesgin.github.io

@mfesgin