Great-LaKeys: An Improved Threshold-PRF and a Novel Exponent-VRF from LWR

Matthias Geihs

Torus Labs
matthias@tor.us

Abstract. Building on the recently proposed LWR-based threshold-PRF LaKey (USENIX'24), we propose two new PRF constructions. First, we present an optimized variant of the LaKey threshold-PRF with significantly reduced round and communication complexity. We achieve this by improving the efficiency of the underlying bit truncation protocol, which may be of independent interest, and by providing tighter parameter bounds. Second, we show that the underlying LWR-based PRF can be proven efficiently in zero-knowledge, and thereby yields a novel and practically efficient exponent-VRF. We provide prototype implementations of both of our constructions and evaluate their concrete performance.

1 Introduction

MPC-friendly PRFs are important building blocks in cryptographic protocols. They enable applications like scalable decentralized key management [GM23] and secure encryption in MPC [GØS$^+$23]. We refer to [GØS$^+$23] for a more comprehensive list of applications.

Moreover, the same PRF constructions are often also useful in zero-knowledge (ZK) protocols, as their structure often also allows for efficient ZK proof generation [AABS$^+$20]. The recently proposed notion of an exponent VRF [BHL24] indeed shows that efficiently provable PRFs are a powerful tool, facilitating efficient threshold key generation and signing protocols.

Here we present two new PRF constructions based on the Learning-With-Rounding (LWR) assumption and inspired by the LaKey PRF design [GM23]: An improved Threshold PRF and a novel, practically efficient Exponent VRF.

1.1 Improved Threshold-PRF

Firstly, we present a variant of the LaKey threshold PRF [GM23] with significantly reduced round and communication complexity. We achieve this mainly by improving the efficiency of the underlying bit truncation protocol. Concretely, we show how previously separate and sequential bit truncation protocols for higher order and lower order bit truncation can be merged into a single protocol that requires the same amount of communication as just the previous higher order bit truncation protocol, essentially cutting communication cost in half. We also improve the analysis of the parameter sizes and thereby are able to further reduce the communication. We provide an open-source implementation of our threshold PRF (tPRF) construction using the `MP-SPDZ` library [Kel20] and compare it with the previous work.

1.2 Novel Exponent-VRF

Building on the LWR-based PRF construction underlying LaKey, we observe that it lends itself well to be turned into an efficient exponent VRF (eVRF), using Bulletproofs [BBB$^+$17] as the proof system. The main observation is that almost all operations of the PRF are linear, and thereby basically free to prove, except for the bit truncations. These, however, we also know how to efficiently prove using Bulletproofs. The resulting eVRF is relatively simple in its construction, supports arbitrary target groups, and is expected to have comparable performance to the recently proposed DDH-based eVRF from [BHL24], which, to the best of our knowledge, does not have an open-source implementation and only works for target groups that have a matching source group (see [BHL24], section 5, *Practical considerations*). We provide an open-source implementation of our eVRF construction using the `zkcrypto/bulletproofs` library [Zer24] and practically demonstrate its performance.

1.3 Organization

We present preliminaries in section 2. Then we present our threshold-PRF optimizations in section 3. Next, we present our novel LWR-based exponent-VRF in section 4. Finally, we evaluate the performance of our constructions in section 5. We also outline some directions for future work in section 6.

2 Preliminaries

2.1 Notation

For two probability distributions X and Y over a finite domain D, we define their statistical distance as

$$\Delta(X,Y) = \frac{1}{2}\sum_{a\in D} |\Pr[X=a] - \Pr[Y=a]| \quad .$$

By $U(D)$ we denote the uniform distribution over a finite domain D.

2.2 LaKey PRF

A *pseudorandom function* (PRF) [GGM86] is an efficiently computable function $F : \mathcal{K}\times\mathcal{X} \to \mathcal{Y}$ such that for a uniform k in $\mathcal{K}$ and a uniform function $f : \mathcal{X} \to \mathcal{Y}$, an oracle for $F(k,\cdot)$ is computationally indistinguishable from an oracle for $f(\cdot)$.

Let $l, m, \bar{q}, \bar{p} \in \mathbb{N}$, where $\bar{q} > \bar{p}$, set $q = 2^{\bar{q}}, p = 2^{\bar{p}}$, and let $H : \{0,1\}^* \to \mathbb{Z}_q^{l\times m}$ be a cryptographic hash function. The LaKey PRF [GM23] is defined as

$$F_{\mathsf{LaKey}} : \mathbb{Z}_q^m \times \{0,1\}^* \to \mathbb{Z}_{\tilde{p}}; (\mathbf{k}, x) \mapsto \mathsf{Acc}(\mathsf{Trunc}(H(x) \cdot \mathbf{k}, \bar{q} - \bar{p}, \bar{q})) \enspace ,$$

where $\mathsf{Trunc} : \mathbb{Z}_q^l \to \mathbb{Z}_p^l; x \mapsto \lfloor (x \bmod q)/(q/p) \rfloor$ truncates the highest and lowest bits (below bit $\bar{q} - \bar{p}$ and above bit $\bar{q}$), and $\mathsf{Acc} : \mathbb{Z}_p^l \to \mathbb{Z}_{\tilde{p}}; (x_1, \ldots, x_l) \mapsto \sum_{i=1}^{l} x_i \cdot p^{i-1}$ maps a vector of integers in $\mathbb{Z}_p$ onto a single integer in $\mathbb{Z}_{\tilde{p}}$.

2.3 Bit truncation in MPC

[CdH10] features two deterministic bit truncation protocols for MPC. The first one is for truncating higher order bits (referred to by Mod2m in their paper, here referred to as TruncHi, shown in Algorithm 1). The second one is for truncating lower order bits (referred to by Trunc in their paper, here referred to as TruncLo, Algorithm 2). Both of these protocols rely on protocols PRandM and BitLT as building blocks, whose functionality and properties we will outline below. In the protocol descriptions, $[a]$ denotes a secret-sharing of a.

Protocol $\mathsf{PRandM}(k, m) \to ([r''], [r'], [r'_{m-1}], \ldots, [r'_0])$ is a protocol that outputs two random secret-shared integers in $[r'']$ and $[r']$, and a vector of secret-shared random bits $[r'_{m-1}], \ldots, [r'_0]$ such that $r'' \in [0, 2^{k+\kappa-m}]$, $r' \in [0, 2^m]$, and $r' = \sum_{i=0}^{m-1} r'_i * 2^i$. Here, κ refers to a globally fixed statistical security parameter. To generate random integers within a certain range, PRandM consumes pre-shared random bits. Concretely, an invocation of $\mathsf{PRandM}(k, m)$ consumes $k + \kappa - m + m = k + \kappa$ pre-shared random bits.

$\mathsf{BitLT}(a, [b_{l-1}], \ldots, [b_0]) \to [c]$ is a protocol that compares a cleartext integer a with a secret-shared integer in binary form $[b_{l-1}], \ldots, [b_0]$ (where $[b_i] \in \{[0], [1]\}$). It outputs $[c] = [1]$ if $a < \sum_{i=0}^{l-1} b_i * 2^i$ and $[c] = [0]$ otherwise. There are various possible implementations of BitLT, but for our purposes we will rely on an implementation that runs in $\log(m)$ online rounds and balances communication with round complexity.

We remark that there is also a constant round version of the comparison protocol at the cost of added communication complexity. However, for our smaller parameter sizes, the round count of the constant-round protocol is the same as the logarithmic round protocol, and for the larger parameter sizes the round count difference is still small (within 1 or 2 rounds).

Algorithm 1 $\mathsf{TruncHi}([a], k, m) \to [a']$

```
1: $([r''], [r'], [r'_{m-1}], \ldots, [r'_0]) \leftarrow \mathsf{PRandM}(k, m)$
2: $c \leftarrow \mathsf{Output}(2^{k-1} + [a] + 2^m[r''] + [r'])$
3: $c' \leftarrow c \bmod 2^m$
4: $[u] \leftarrow \mathsf{BitLT}(c', [r'_{m-1}], \ldots, [r'_0])$
5: $[a'] \leftarrow c' - [r'] + 2^m[u]$
6: return $[a']$
```

Algorithm 2 $\mathsf{TruncLo}([a], k, m) \to [a'']$

```
1: $[a'] \leftarrow \mathsf{TruncHi}([a], k, m)$;
2: $[a''] \leftarrow ([a] - [a'])/2^m$;
3: return $[a'']$;
```

2.4 Exponent VRFs

An exponent VRF [BHL24] is defined by a set of algorithms $(\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Verify})$, associated with a cyclic group generated by a group element G, with the following properties.

$\mathsf{KeyGen}() \to (k, vk)$**:** The key generation algorithm randomly generates a secret evaluation key k and a public verification key vk.

$\mathsf{Eval}(k, x) \to (y, Y, \pi)$**:** On input an evaluation key k, and a message x, the evaluation algorithm outputs (y, Y, π), where y is a pseudorandom value in the target domain, $Y = G * y$, and π is a proof for the correct evaluation.

$\mathsf{Verify}(vk, x, Y, \pi) \to \{0, 1\}$**:** On input a verification key vk, a message x, an exponent output Y, and a proof π, the verification algorithm outputs 1 if π certifies that Y is the correct output for input x, and 0 otherwise.

In addition, an exponent VRF must satisfy consistency, pseudo-randomness, and simulatable verifiability. For consistency, we require that Eval yields consistent outputs (i.e., $Y = G * y$). For pseudo-randomness, we require that $(\mathsf{KeyGen}, \mathsf{Eval})$ is a secure PRF (w.r.t. the first output y). For simulatable verifiability, we require that one cannot generate two conflicting proofs for the same input x (i.e., $(vk, x, Y_0, \pi_0, Y_1, \pi_1)$ such that $\mathsf{Verify}(vk, x, Y_0, \pi_0) = \mathsf{Verify}(vk, x, Y_1, \pi_1) = 1$ and $Y_0 \neq Y_1$) and that a proof does not leak anything beyond the correctness of the statement. We refer to [BHL24], Section 3, for a more formal definition.

2.5 Bulletproofs

Bulletproofs [BBB+17] is an efficient zero-knowledge proof system based on elliptic-curve cryptography. In short, it allows a prover to efficiently generate a short proof for an arbitrary computation (e.g., expressed as a Rank-1 Constraint System or R1CS) such that a verifier can verify the correctness of the computation much more efficiently than recomputing it themselves. Moreover, the prover can choose to hide some of the inputs of the computation but still convince the verifier that it knows corresponding inputs that satisfy certain constraints. For example, a prover could convince the verifier that it knows the secret key belonging a public key without revealing anything other than the truthfulness of that statement.

We will use the following algorithmic notation when working with Bulletproofs.

$\mathsf{Bul.Setup}() \to p_{\mathsf{Bul}}$**:** The setup algorithm outputs public parameters p_{Bul} that are used in the proof system, where by $p_{\mathsf{Bul}}.G$ we denote the group generator used for commitment generation.

$\mathsf{Bul.Com}(p_{\mathsf{Bul}}, k, d) \to K$**:** The commitment algorithm generates a commitment K, which is a group element, to a secret k, which is a group scalar, using blinding value d.

$\mathsf{Bul.Prove}(p_{\mathsf{Bul}}, C, X, W) \to \pi$**:** The prove algorithm generates a proof π for a constraint system C to be satisfied by inputs X and witness W.

$\mathsf{Bul.Verify}(p_{\mathsf{Bul}}, C, X, \pi) \to \{0,1\}$**:** The verify algorithm checks the correctness of a proof π for a constraint system C and inputs X.

3 Improvements to Threshold-PRF

We present our two improvements to the LaKey threshold PRF construction. The first one is an improved bit truncation protocol, which lies at the heart of LaKey and significantly reduces rounds and communication. The second one is a tighter bound on the statistical distance between the intermediate LWR-based PRF output and the target distribution, which allows us to choose smaller parameters and further reduces the computation and communication requirements of LaKey. We will evaluate the impact of these improvements in section 5.

3.1 Bit truncation with fewer rounds and communication

We present our improved bit truncation protocol in Algorithm 3. The main observation is that the existing protocols TruncHi and TruncLo (subsection 2.3) can be merged into a single protocol TruncHiLo that enjoys the same round and communication complexity as just running TruncHi.

Algorithm 3 $\mathsf{TruncHiLo}([a], k, m, n) \to [a']$

```
1: $([r''], [r'], [r'_{n-1}], \dots, [r'_0]) \leftarrow \mathsf{PRandM}(k, n)$
2: $c \leftarrow \mathsf{Output}(2^{k-1} + [a] + 2^n[r''] + [r'])$ ▷ 1 round
3: for $x \in \{m, n\}$ do ▷ Can run in parallel.
4:     $c' \leftarrow c \bmod 2^x$
5:     $[u] \leftarrow \mathsf{BitLT}(c', [r'_{x-1}], \dots, [r'_0])$ ▷ $\log(x)$ rounds
6:     $[r'] \leftarrow \sum_{j=0}^{x-1} 2^j [r'_j]$
7:     $[a'_x] \leftarrow c' - [r'] + 2^x[u]$
8: $[a'] \leftarrow ([a'_n] - [a'_m])/2^m$
9: return $[a']$
```

Correctness. We need to show that for all valid $([a], k, m, n)$,

$$\mathsf{TruncHiLo}([a], k, m, n) = \lfloor (a \bmod 2^n)/2^m \rfloor \ .$$

Let $[a'] = \mathsf{TruncHiLo}([a], k, m)$. Observe that $[a'] = ([a'_n] - [a'_m])/2^m$ by the construction of TruncHiLo, where $[a'_n] = [a] \bmod 2^n$ and $[a'_m] = [a] \bmod 2^m$ by the construction of TruncHi. Finally, we observe that we have

$$(([a] \bmod 2^n) - ([a] \bmod 2^m))/2^m = \lfloor ([a] \bmod 2^n)/2^m \rfloor \ ,$$

because $n \geq m$ and therefore $([a] \bmod 2^n) - ([a] \bmod 2^m)$ is divisible by 2^m.

Security. We need to show that the protocol does not leak anything about the secret value a. Our analysis follows the analysis of [CdH10], Section 2, *Security*. The only value that is revealed in the protocol is $c = 2^{k-1} + [a] + 2^n[r''] + [r']$. We observe that $a \in \mathbb{Z}_{2^k}$ and that $r = 2^n[r''] + [r']$ is a random element in $\mathbb{Z}_{2^{k+\kappa}}$. By Theorem 2 of [CdH10] it follows that $a + r$ is κ-statistically-close to the uniform distribution over $\mathbb{Z}_{2^{k+\kappa}}$.

Efficiency. Running $\mathsf{TruncHiLo}([a], k, m, n)$ takes $\log(n) + 1$ online rounds and consumes $k + \kappa$ pre-shared random bits.

3.2 Exact bound on statistical distance to target distribution

For LaKey to be pseudorandom over $\mathbb{Z}_{\tilde{p}}$, we require that the statistical distance between $\mathsf{Acc}(U(\mathbb{Z}_p^l)) \bmod \tilde{p}$ and $U(\mathbb{Z}_{\tilde{p}})$ is negligible. In [GM23], it was established that the distance is bounded by $\tilde{p}/p^l$. The following Lemma 1 establishes a tighter bound and thereby allows for choosing smaller l.

Lemma 1. *For $m, n \in \mathbb{N}$,*

$$\Delta(U(\mathbb{Z}_m), U(\mathbb{Z}_n) \bmod m) = \left| \frac{(n \bmod m)^2 - (n \bmod m)m}{nm} \right| .$$

Proof. Writing out the definition, we have

$$\Delta(X, Y) = \frac{1}{2} \sum_{a \in \mathbb{Z}_m} \left| \Pr[X = a] - \Pr[Y = a] \right| .$$

For $a \in \mathbb{Z}_m$, we have $\Pr[X = a] = 1/m$ and

$$\Pr[Y = a] = \begin{cases} \frac{n-(n \bmod m)}{nm} & \text{if } a \geq n \bmod m, \\ \frac{n-(n \bmod m)+m}{nm} & \text{if } a < n \bmod m. \end{cases}$$

Denote $r = (n \bmod m)$. It follows that

$$\begin{aligned} \Delta(X, Y) &= \frac{1}{2}\left(\left(\sum_{a \in \mathbb{Z}_m : a < r} \left| \frac{1}{m} - \frac{n-r+m}{nm} \right| \right) + \left(\sum_{a \in \mathbb{Z}_m : a \geq r} \left| \frac{1}{m} - \frac{n-r}{nm} \right| \right)\right) \\ &= \frac{1}{2}\left(r \left| \frac{1}{m} - \frac{n-r+m}{nm} \right| + (m-r) \left| \frac{1}{m} - \frac{n-r}{nm} \right| \right) \\ &= \left| \frac{r^2 - rm}{nm} \right| . \end{aligned}$$

□

4 Novel Exponent-VRF

We present our novel exponent VRF eLaKey in Definition 1. It is constructed by composing the LaKey PRF together with an efficient proof system for its correct evaluation, which we realize using Bulletproofs.

Definition 1 (LaKey-eVRF). *Let* LaKey *denote the LaKey PRF (subsection 2.2) and let* Bul *denote the BulletProofs proof system (subsection 2.5). Given* $(p_{\mathsf{Bul}}, K, x, Y; k_{\mathsf{PRF}})$*, where* $K = (K_1, \ldots, K_m)$ *and* $k_{\mathsf{PRF}} = (k_1, \ldots, k_m)$*, let* C_{LaKey} *be a constraint system for the following constraints:*

- $\forall i \in \{1, \ldots, m\} : K_i = \mathsf{Bul.Com}(p_{\mathsf{Bul}}, k_i, 0)$
- $y = \mathsf{LaKey.Eval}(k_{\mathsf{PRF}}, x)$
- $Y = p_{\mathsf{Bul}}.G * y$

We refer to subsection 4.2 for a description of an efficient realization of C_{LaKey}*. The* eLaKey *exponent-VRF is defined as follows.*

```
eLaKey.KeyGen() → $(k, vk)$.
    – Compute $k_{\mathsf{PRF}} \leftarrow \mathsf{LaKey.KeyGen}()$ and $p_{\mathsf{Bul}} \leftarrow \mathsf{Bul.Setup}()$.
    – For $i \in [\mathsf{LaKey}.m]$, compute $K_i \leftarrow \mathsf{Bul.Com}(p_{\mathsf{Bul}}, k_{\mathsf{PRF},i}, 0)$.
    – Set $K \leftarrow (K_1, \ldots, K_m)$.
    – Set $k \leftarrow (k_{\mathsf{PRF}}, p_{\mathsf{Bul}})$ and $vk \leftarrow (K, p_{\mathsf{Bul}})$.
eLaKey.Eval$(k, x) \rightarrow (y, Y, \pi)$.
    – Parse $(k_{\mathsf{PRF}}, p_{\mathsf{Bul}}) \leftarrow k$.
    – Compute $y \leftarrow \mathsf{LaKey.Eval}(k_{\mathsf{PRF}}, x)$ and $Y \leftarrow p_{\mathsf{Bul}}.G * y$.
    – Compute $\pi \leftarrow \mathsf{Bul.Prove}(p_{\mathsf{Bul}}, C_{\mathsf{LaKey}}, [K, x, Y], k_{\mathsf{PRF}})$.
eLaKey.Verify$(vk, x, Y, \pi) \rightarrow \{0, 1\}$.
    – Parse $(K, p_{\mathsf{Bul}}) \leftarrow vk$.
    – Return $\mathsf{Bul.Verify}(p_{\mathsf{Bul}}, C_{\mathsf{LaKey}}, [K, x, Y], \pi)$.
```

4.1 Security

In the following we show that eLaKey satisfies consistency, pseudo-randomness, and simulatable verifiability.

For consistency it suffices to show that for all k, x and $(y, Y, \pi) \leftarrow \mathsf{Eval}(k, x)$, we have $Y = G * y$. This is obviously the case as Y is computed as $Y \leftarrow G * y$.

For pseudo-randomness, we need to show that $(\mathsf{KeyGen}, \mathsf{Eval})$ is a secure PRF when restricted to ouput y. This also immediately follows from the fact that y is computed as $y \leftarrow \mathsf{LaKey.Eval}(k_{\mathsf{PRF}}, x)$ and assuming LaKey is a secure PRF.

For simulatable verifiability, we need to show that eLaKey is verifiable and simulatable. In other words, we need to show that one cannot come up with two conflicting proofs, and that an honestly generated proof does not leak anything beyond the correctness of the statement. Verifiablity and simulatability immediately follow from the fact that π is a Bulletproofs proof for the relation C_{LaKey} and that the Bulletproofs proof system Bul is statistical zero-knowledge and computationally sound ([BBB+17], Theorem 5).

4.2 Efficient realization of constraint system

We now describe how the constraint system C_{LaKey} can be efficiently realized as a rank-1 constraint system (R1CS). The main observation is that most of the operations of $\mathsf{LaKey.Eval}$ are linear, except for the bit truncation. Bit truncation, however, can be efficiently realized via bit decomposition, which we know can be efficiently represented as an R1CS ([BBB+17], Section 4). Figure 1 describes the C_{LaKey} constraint system in more detail. Here, $\mathsf{BitDecomp}$ denotes an algorithm that on input $(X; x)$, where $X = (X_1, \ldots, X_m)$ is a commitment vector and $x = (x_1, \ldots, x_m)$ is a witness vector such that X_i is a commitment to x_i, outputs (C, B), where C is a constraint system and B is a commitment matrix, such that each row i of B is a commitment vector to a bit decomposition of x_i. We refer to subsection 5.2 for a concrete implementation of this constraint system.

Input: $p_{\mathsf{Bul}}; K = (K_1, \ldots, K_m), x, Y; k_{\mathsf{PRF}} = (k_1, \ldots, k_m)$.
Output: Constraints for the following relation:
- $\forall i \in \{1, \ldots, m\}$: $K_i = \mathsf{Bul.Com}(p_{\mathsf{Bul}}, k_i, 0)$
- $Y = p_{\mathsf{Bul}}.G * \mathsf{LaKey.Eval}(k_{\mathsf{PRF}}, x)$

Procedure:
1. Set $C_1 = (\forall i \in \{1, \ldots, m\} : K_i = \mathsf{Bul.Com}(p_{\mathsf{Bul}}; k_i, 0))$.
2. Let $A = \mathsf{LaKey}.H(x)$ and set $(C_2, Y_2) \leftarrow \mathsf{BitDecomp}(A * K; A * k_{\mathsf{PRF}})$.
3. Let $Y_3 = \mathsf{LaKey.Acc}(\mathsf{LaKey.Trunc}(Y_2))$ and set $C_3 \leftarrow (Y = Y_3)$.
4. Output C_1, C_2, C_3.

Fig. 1. R1CS for eLaKey.

4.3 Comparison with eVRFs from [BHL24]

[BHL24] proposes two eVRF constructions, one based on DDH and another one based on Paillier. Both of their constructions require some tricks to make them work for practical use cases.

For their DDH-based eVRF, they describe how to instantiate the eVRF when the target group $\mathbb{G}_T$ is either `secp256k1` or `ed25519`. For the case of `secp256k1`, the target group needs to be the standard group of points on curve $y^2 = x^3 + 7$ over some prime field $\mathbb{F}_p$. To find a corresponding source group, they observe that, due to a theorem by Silverman and Stange [SS11], curve $y^2 = x^3 + 7$ over $\mathbb{F}_q$, for some q, has prime order p, and can therefore be used as the source group. A different observation is made to find a corresponding source group for the target group `ed25519`. We remark that no such tricks are required for instantiating our eVRF eLaKey as we do not need to find a matching source group.

For their Paillier-based eVRF, they remark that efficient range proofs for Paillier require a trusted setup, which is a relatively strong assumption in itself that rules out certain pratical use cases where such a setup procedure is not feasible. Again, our eVRF eLaKey does not have such a limitation.

5 Evaluation

5.1 Threshold-PRF

We implemented our improved threshold PRF LaKey2 using MP-SPDZ [Kel20]. The code is available at github.com/torusresearch/lakey-tprf. As in [GM23], we target a 256-bit prime field and fix the lattice dimension to 512. As modulo parameter sets we use ($\bar{q} = 12, \bar{p} = 8$) and ($\bar{q} = 32, \bar{p} = 24$). The corresponding scheme instances are denoted as LaKey_{12} and $\mathsf{LaKey2}_{12}$, and LaKey_{32} and $\mathsf{LaKey2}_{32}$, respectively. Due to our improved lower bound on the statistical distance of the output distribution, we are able to reduce the size of the interal parameter l from $l = 37$ for LaKey_{12} and $l = 13$ for LaKey_{32} to $l = 32$ for $\mathsf{LaKey2}_{12}$ and $l = 11$ for $\mathsf{LaKey2}_{32}$, which in turn reduces the number of LWR instances we need to evaluate in parallel and thereby reduces computation as well as communication costs.

Table 1 gives an overview of the performance measurements for LaKey2 compared with LaKey. Here, the parameters with a subscript K correspond to the communication-optimized LaKey variant denoted by OPT in [GM23]. The measurements have been performed on a machine with a 10-core M1 Pro CPU and 32 GB RAM, simulating 3 parties with a reconstruction threshold of 2, and using protocol `mal-shamir`.

We observe that LaKey2 significantly outperforms LaKey on all measures. Computation is reduced by 12% to 17%. Online communication is reduced by 14% to 16% and total communication by 35% to 41% in case of the regular construction and by 15% to 16% in case of the communication-optimized construction. Round count is reduced significantly to 5 or 6 rounds, depending on the parameter choice, from previously 8 or 10, respectively. The reduction of the number of required pre-shared random bits is reflected by the lower communication cost. We also show the difference in key sizes between the regular and the communication-optimized variant of LaKey. Due to the much improved communication complexity of LaKey2, the communication-optimized variant can almost be considered obsolete due to the much larger key size.

5.2 Exponent-VRF

We implemented our novel exponent VRF eLaKey using the `bulletproofs` Rust library [Zer24]. As scheme parameters we use $m = 512$, $\bar{q} = 12$, and $\bar{p} = 8$, and we target a 255-bit prime field, concretely the scalar field of `curve25519`. The source code can be found at github.com/torusresearch/lakey-evrf.

Benchmarks are run on a machine with a 10-core M1 Pro CPU and 32 GB RAM. The results can be found in Table 2. We observe that key generation and verification are both well below 100ms. Evaluation is a bit more costly and takes

Table 1. Comparison of LaKey and LaKey2 threshold PRFs.

Protocol	**Computation** (ms)	**Communication** (MB, online/total)	**Rounds**	**Random Bits**	**Key Size** (B)
		LaKey			
LaKey_{12}	11.34	0.41 / 6.01	8	4625	768
LaKey_{32}	8.06	0.43 / 3.85	10	2405	768
$\mathsf{LaKey}_{\mathsf{K}12}$	24.79	0.33 / 3.85	8	2753	28416
$\mathsf{LaKey}_{\mathsf{K}32}$	12.86	0.36 / 2.79	10	1541	28416
		LaKey2			
$\mathsf{LaKey2}_{12}$	9.80	0.35 / 3.56	5	2336	768
$\mathsf{LaKey2}_{32}$	6.68	0.36 / 2.49	6	1243	768
$\mathsf{LaKey2}_{\mathsf{K}12}$	21.84	0.29 / 3.29	5	2336	24576
$\mathsf{LaKey2}_{\mathsf{K}32}$	10.78	0.31 / 2.33	6	1243	24576

around 260ms. The proof size is around 1 kB. We expect that these figures can further be tweaked by experimenting with different parameter sets. Overall, we expect these figures to suffice for many practical applications, like distributed key generation and threshold signing.

Comparison to [BHL24]. In comparison to the eVRFs proposed in [BHL24], we observe the following. For their DDH-based eVRF, the authors estimate a proof size of about 900 bytes, and proving and verification times of just a few milliseconds, while the number of constraints is between 1282 and 1537. For their Pallier-based eVRF, the authors estimate the proving time at about 375ms and the verification time at about 168ms. They do not make any statements about the proof size. We note that, to the best of our knowledge, both of their proposed eVRFs have not been implemented and the provided numbers are only estimates.

Table 2. Exponent-VRF eLaKey benchmark results and estimations from [BHL24]. A question mark (?) means that the information has not been provided in the respective publication. A dash (-) means that the measure is not applicable.

Scheme	eLaKey (measured)	**DDH-[BHL24]** (estimated)	**Paillier-[BHL24]** (estimated)
Key generation	74ms	?	?
Evaluation	263ms	∼5-14ms	∼375ms
Verification	27ms	∼5-14ms	∼168ms
Proof size	1121B	∼900B	?
Constraints	1056	∼1282-1537	-

6 Open questions

We foresee at least two interesting future lines of work.

The first question is whether we can also build an oblivious PRF from the same underlying PRF construction. The main obstacle here is that the LaKey threshold PRF as a first step requires each protocol participants to map the public input x onto a random matrix A in $\mathbb{Z}_q^{l\times m}$. In our construction, we use a random oracle for that mapping which we can instantiate using a hash function like SHA2 or SHA3. If the input is secret, as required for an OPRF, we would need to find an efficient way to compute this mapping in MPC.

The second question is whether our underlying PRF construction can be efficiently evaluated in FHE. We have observed that the construction lends itself well to be run in MPC as well as in ZK proof systems, but while FHE systems share some similarities, it is currently unclear what the concrete efficiency of an implementation in FHE would be. Such a construction would be interesting, as it could also be the basis for an OPRF construction, as shown in [ADD+23], with the benefit that the security of our PRF is reducible to standard LWE. First steps in this direction have been made in [DJL+24].

Acknowledgements

Thanks to Andreas Erwig for proposing the idea to investigate LaKey further in the context of exponent-VRFs, and thanks to Hart Montgomery for his continued guidance around the topic of LWR-based PRFs.

References

AABS+20. A. Aly, T. Ashur, E. Ben-Sasson, S. Dhooghe, and A. Szepieniec. Design of symmetric-key primitives for advanced cryptographic protocols. *IACR Transactions on Symmetric Cryptology*, 2020(3):1–45, Sep. 2020.

ADD+23. M. R. Albrecht, A. Davidson, A. Deo, and D. Gardham. Crypto dark matter on the torus: Oblivious prfs from shallow prfs and fhe. Cryptology ePrint Archive, Paper 2023/232, 2023. `https://eprint.iacr.org/2023/232`.

BBB+17. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. Cryptology ePrint Archive, Paper 2017/1066, 2017. `https://eprint.iacr.org/2017/1066`.

BHL24. D. Boneh, I. Haitner, and Y. Lindell. Exponent-VRFs and their applications. Cryptology ePrint Archive, Paper 2024/397, 2024. `https://eprint.iacr.org/2024/397`.

CdH10. O. Catrina and S. de Hoogh. Improved primitives for secure multiparty integer computation. In J. A. Garay and R. D. Prisco, editors, *Security and Cryptography for Networks, 7th International Conference, SCN 2010, Amalfi, Italy, September 13-15, 2010. Proceedings*, volume 6280 of *Lecture Notes in Computer Science*, pages 182–199. Springer, 2010.

DJL$^+$24. A. Deo, M. Joye, B. Libert, B. R. Curtis, and M. de Bellabre. Homomorphic evaluation of LWR-based PRFs and application to transciphering. Cryptology ePrint Archive, Paper 2024/665, 2024. `https://eprint.iacr.org/2024/665`.

GGM86. O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 34(4):792–807, 1986.

GM23. M. Geihs and H. Montgomery. Lakey: Efficient lattice-based distributed prfs enable scalable distributed key management. Cryptology ePrint Archive, Paper 2023/1254, 2023. `https://eprint.iacr.org/2023/1254`.

GØS$^+$23. L. Grassi, M. Øygarden, M. Schofnegger, and R. Walch. From farfalle to megafono via ciminion: The PRF hydra for MPC applications. In C. Hazay and M. Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part IV*, volume 14007 of *Lecture Notes in Computer Science*, pages 255–286. Springer, 2023.

Kel20. M. Keller. MP-SPDZ: A versatile framework for multi-party computation. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020.

SS11. J. H. Silverman and K. E. Stange. Amicable pairs and aliquot cycles for elliptic curves. *Experimental Mathematics*, 20(3):329–357, 2011.

Zer24. Zero-knowledge Cryptography in Rust. Bulletproofs. `https://github.com/zkcrypto/bulletproofs`, 2024.