Faster Proofs and VRFs from Isogenies

Shai Levin[1] and Robi Pedersen[2,3]

[1] University of Auckland, New Zealand
shai.levin@auckland.ac.nz
[2] COSIC KU Leuven, Belgium
[3] Technical University of Denmark
robi.pedersen@protonmail.com

Abstract. We improve recent generic proof systems for isogeny knowledge by Cong, Lai, Levin [27] based on circuit satisfiability, by using radical isogeny descriptions [20,21] to prove a path in the underlying isogeny graph. We then present a new generic construction for a verifiable random function (VRF) based on a one-more type hardness assumption and zero-knowledge proofs. We argue that isogenies fit the constraints of our construction and instantiate the VRF with a CGL walk [23] and our new R1CS system. As a different contribution, we also propose a new VRF in the effective group action description of isogenies from [1]. Our protocol takes a novel approach based on the polynomial-in-the-exponent technique first described in [36], but without the need of a trusted setup or heavy preprocessing. We compare our protocols to the current state-of-the-art isogeny VRFs by Leroux [56] and Lai [54], with a particular emphasis on computational efficiency.

Keywords: Isogeny-based cryptography · Verifiable Random Function · Cryptographic Protocols · Zero-Knowledge Proofs

1 Introduction

Proofs of isogeny knowledge. Isogenies are homomorphisms between elliptic curves. Given two elliptic curves it is assumed to be hard to find an isogeny connecting them, even for quantum computers. Proving knowledge of an isogeny connecting two given elliptic curves has been an active field of study since the inception of isogeny-based cryptography, as it allows to realize important constructions, such as ID protocols and signatures [6, 13, 29, 31, 33–35]. Proofs of isogeny knowledge also allow parties to prove that isogenies have been computed in a given, predetermined way. For instance, proving that one has indeed walked starting from a given elliptic curve, allows the distributed computation of elliptic curves of unknown endomorphism rings [8, 58], while showing that one has used a previously determined seed to construct an isogeny allows the realization of efficient distributed key generation schemes [4, 5]. The latter proofs can be extended to showing that two isogenies are parallel, which allows the construction of oblivious pseudo-random functions [7, 16]. A recent work by Cong, Lai and

Levin [27] is the first to have explored building proofs of isogeny knowledge from circuit satisfiability conditions. In particular, the authors encode isogeny walks in a graph using rank-one constraint system (R1CS) and achieve particularly fast proofs of isogeny knowledge, when compared to previous techniques, such as [8,31].

Verifiable random functions. *Verifiable Random Functions* (VRFs) are a cryptographic primitive which allows an evaluator of a pseudorandom function to prove the correctness of their output. A VRF is instantiated with a public-private key pair (pk, sk), and on the input of a message m and private key sk, outputs a pseudorandom value h and proof π. A verifier may then take (pk, m, h, π) and accept or reject. VRFs must satisfy *residual pseudorandomness* (a related but distinct notion compared to the *pseudorandomness* of a PRF) which states that for a new message m, the output h is indistinguishable from random, even if an adversary has already seen evaluations and proofs for arbitrary, distinct messages. VRFs also satisfy *unique provability*, which means that two distinct evaluations of the VRF under the same message and secret key cannot both have accepting proofs. First introduced in [57], VRFs have found applications in blockchain consensus, such as in Algorand [24], where a VRF is used after each block in a *lottery* to determine who forms the next block; distributed randomness beacons [26,65], where publicly generated randomness can be generated in a distributed, trustless computation; and DNSSEC [47].

Post-quantum VRFs. Until recently, VRF constructions have been based exclusively on classical assumptions which require the hardness of computing discrete-logarithms or integer factoring, and thus are not post-quantum secure. Hence, it is of interest to propose candidates for post-quantum VRFs. Post-quantum secure VRFs have been proposed from lattice [41,42] and hash [19,40] assumptions. We note that the constructions in [40,41] are *short-term* VRFs, which trade off long-term usability for higher efficiency. Another short-term VRF has been proposed in [19], but was broken in [15]. More recently, the first post-quantum VRFs from isogenies have been proposed by Lai [54] and Leroux [56]. Lai's construction is based on a Naor-Reingold type PRF [59] in the effective group action setting [1] and introduces new proof systems to realize verifiability. In contrast, Leroux's VRF evaluations are large prime degree isogenies computed via the Deuring correspondence and proven correct by providing a higher dimensional representation of that isogeny. The latter results in the smallest proof sizes in the post-quantum setting (cf. [56, Table 1]).

Contributions

In this work, we optimise R1CS circuits in order to prove isogeny walks using generic proof-systems. We further expand the description to allow proofs of more complex statements. We then present a verifiable random function from isogenies, one using a CGL-type walk [23] in the isogeny graph and our new proof systems. In addition, we present a second novel VRF in the group action setting of isogenies. We start this paper in Section 2 with some background on isogenies in both the generic and the group action setting, then introduce zero-knowledge proof systems, including zkSNARKs and verifiable random functions. Our main contributions can be summarized as follows.

Optimised generic proof descriptions. In Section 3, we introduce new descriptions for generic proof systems from isogenies using R1CS. This line of research was initiated by [27], which built a circuit description using modular polynomials that could be plugged into post-quantum zkSNARKs, such as Aurora [11] and Ligero [2], and result in very fast proofs of isogeny knowledge. We improve upon their work by designing a different isogeny description using the radical isogeny formulae from [20, 21]. Rather than having quadratic and cubic terms, the radical isogeny formulas are linear and allow a much more compact description in the R1CS setting when compared to [27], effectively reducing the number of constraints in a non-backtracking walk of length n in the 2-isogeny graph from $12n+3$ down to $7n$ constraints over $\mathbb{F}_p \times \mathbb{F}_p$. Concurrently to our work, the authors of [52] construct R1CS instances via canonical modular polynomials in order to prove knowledge of isogeny paths. For paths of 2-isogenies, their number of constraints is roughly comparable to our radical isogeny approach.

We further extend these circuit descriptions to also allow the proof of a specific CGL walk given a predetermined (secret) input bitstring, as well as to show that two given isogeny walks have been performed using the same input seed. In order for the R1CS circuit to distinguish square roots, and thus force canonical directions of each step of the walk, we require that our R1CS circuit be evaluated via $\mathbb{F}_p$-arithmetic where $p \equiv 3 \pmod 4$. We note that many post-quantum SNARKs are based on the low degree test FRI [10] (or its many variants), which require operations over a highly 2-adic field. Fortunately, there have been recent advances in post-quantum field-agnostic SNARKs, such as Brakedown and its optimisations [14, 49, 50], Orion [67] and BaseFold [68] on which we can base our new proof system.

CGL-based VRF. We start Section 4 by introducing *unpredictable functions*, which are functions that satisfy a given one-more type hardness assumption that we describe in Problem 4. We prove that these functions, together with the typical properties of non-interactive zero-knowledge proof systems, are enough to build a secure VRF. We go on by instantiating unpredictable functions with walks in isogeny graphs. In particular, we present a VRF construction from a CGL-type walk in the 2-isogeny graph. Using the proof system from Section 3, that two isogeny walks have been computed using the same seed, we can add verifiability to our scheme. The result is a very fast protocol in terms of evaluation and verification cost. A recent VRF from isogenies has also been proposed by Leroux [56], which uses the interplay of the Deuring correspondence with higher dimensional isogenies to prove and verify the correct output. The suggested protocol is based on a new one-more type assumption and leads to particularly small proof sizes. In contrast, using [14] as the underlying proof system, our

VRF evaluation time is expected to be much faster than the one in [56], while the verification time is comparable. Furthermore, our protocol enjoys very small public key sizes. On the downside, our proof size is considerably larger, but strongly depends on the SNARK used. While [14] still yields large proof sizes in our setting, the state-of-the-art has developed significantly even within the past 12 months. We expect that further improvements to expander code multilinear commitments will continue to improve estimated proof sizes. At present our protocol requires a starting curve of unknown endomorphism ring, obtained via a trusted setup ceremony such as [8]. VRFs are typically applied in the context of multi-party protocols, where this ceremony may be performed in the setup phase. However, it is likely that parameters may be constructed such that the protocol may achieve full unique provability without the need for a trusted setup, as discussed 4.2.1.

Group action based VRF. In Section 5, we also present a new isogeny-based VRF in the effective group action setting. Inspired by the recent group action based OPRF [36], we construct a VRF with output $[f(m)]E_0$, where $f(X)$ is a secret polynomial held by the server and m the client's input message. We design new NIZKs to prove, that a given evaluation of the VRF indeed encodes a polynomial of the correct degree that also defines the server's public key. Our approaches for both the VRF construction and the proof are novel, and outperform the previous state of the art constructions by Lai [54] by up to one to two orders of magnitude in terms of proof size, proof cost and public key size (see Table 2).

Performance comparison to existing works. We remark that our protocol from Section 4 is a blueprint for a protocol. Serving as a proof of concept, the blueprint is the first of its kind as a VRF based on generic, succinct proof systems (and indeed the first proposal for an isogeny-based protocol which applies generic proof systems). Succinct proof systems are challenging to estimate performance for, especially in the case of isogeny computations, which operate over large prime fields that require big integer arithmetic not natively supported by proof system implementations. Since this would require substantial engineering effort and great care in order to support zero-knowledge and the large field arithmetic, we leave concrete implementation details and more accurate performance evaluations to future work.

Nevertheless, we provide a speculative comparison of our VRFs at NIST level 1 security with the current state-of-the-art post-quantum (long-term) VRFs in Table 1 below. Our VRF construction has particularly low public key sizes, and fast proof and verification times. We note that in the context of VRFs to blockchain consensus and randomness beacons, verification time may be prioritised over other performance metrics, since, for a single generated evaluation, verification is typically performed by multiple parties. Although the benchmarks have not been computed using the same hardware, we get an idea of the relative costs of the different VRFs. In particular, we can see that our fast parameter evaluation time is much lower than SL-VRF or DeuringVRF. We note that the lattice-based LaV construction from [42] doesn't provide an estimate of their runtimes, which makes a direct comparison hard. For the verification, we find a factor 10 improvement with regards to SL-VRF, but are about a factor 1.75× slower compared to DeuringVRF.

Remark 1. The area of field agnostic SNARKs is moving quickly. As of the time of writing, the current state of the art is [14], which outperforms Basefold, Brakedown and Brakedown-improved [50]. We expect that performance of field agnostic SNARKs will continue to improve in the near future. Already, the authors of [14] claim that the optimisations of [37] yield a $\sqrt{2}\times$ improvement to proof size, and a 2× improvement to proof and verification time. If this is the case, then our proofs will already outcompete the state of the art in all metrics except proof size.

In Table 2, we further compare our new group action based VRF with the current state-of-the-art constructions Capybara and Tsubaki [54]. These constructions are instantiated with the effective group action from [13] and achieve 128-bit classical security, but fall short of NIST-level 1 [17, 62]. In the table, it is clearly visible that our construction outperforms [54] in terms of computation and communication cost. Furthermore, it also relies on the computational assumption from Problem 5, rather than on the (square) decisional Diffie-Hellman.

	T_{KeyGen}	T_{Eval}	T_{Verif}	$\lvert\text{PK}\rvert$	$\lvert\text{Output}\rvert$	Assumption
SL-VRF [19]	0.3 ms	765 ms	475 ms	48 B	40 kB	Hash (LowMC)
LaV [42]	?	?	?	9 kB	10 kB	Lattice (MSIS/MLWR)
DeuringVRF [56]	127 ms	174 ms	20 ms	192 B	256 B	Isogeny (OMIP)
Ours (Sec. 4)	5 ms	50 ms	35 ms	64 B	430 kB	Isogeny (Problem 4)

Table 1: Comparison of our CGL-based VRF from Section 4 with other post-quantum VRFs. We compare times for key generation, evaluation and verification of the VRFs, as well as public key and total output size. We estimate performance of our parameters using the reported results of [14] for a circuit of size 2^{13} and a 256-bit prime, and estimating a 256-bit input CGL evaluation as 5 ms[4].

[4] Derived from the rust implementation `https://github.com/levanin/cgl_rs` which uses the less efficient modular polynomial approach.

	T^{GA}_{Keygen}	T^{GA}_{Eval}	|PK|	|Output|	Assumption
Capybara [54]	1	109500	8.3 kB	39 kB	DDH
Tsubaki [54]	1	69255	5.3 kB	34 kB	SDDH
$\deg f = 1$	1	163	1.2 kB	6.4 kB	Problem 5
$\deg f = 2$	2	244	2.3 kB	9.0 kB	
$\deg f = 3$	2	244	2.4 kB	11.6 kB	

Table 2: Comparison of our group action VRF from Section 5 with the previous state-of-the-art by Lai [54] at classical security level $\lambda = 128$. Our VRF takes the degree of a polynomial $f(X)$ as input and different degrees imply different security guarantees as discussed in Section 5. We estimate performance in terms of the number of group actions performed. Our protocol is performed over a prime of $\log p \approx 12\lambda$, whereas [54] is over the CISDH-512 prime. This is not a direct comparison, since we perform group actions in the KLaPoTi framework [61, Algorithm 2], but as shown, we require several orders of magnitude less evaluations and still have smaller public keys. Furthermore, the computational cost of [54] scales quadratically in the security parameter, while our scheme does so linearly (cf. Sec. 5.2). We note that verification times are only one group action less than evaluation times and are therefore omitted.

2 Background

2.1 Elliptic curves and isogenies

In our work we consider elliptic curves of the form $E : y^2 = x^3 + ax^2 + cx$ where $c \neq 0$ and $a^2 - 4c \neq 0$. The *j-invariant* of E is expressed as $j(E) := 256 \cdot \frac{(a^2-3c)^3}{c^2(a^2-4c)}$. Two elliptic curves E, E' defined over a field $\mathbb{F}$, are isomorphic if and only if $j(E) = j(E')$. An *isogeny* is a surjective map between elliptic curves of finite kernel, which acts as a group homomorphism between their group of points over a field. An isogeny ϕ is *separable* if it is determined solely by its kernel, denoted $\ker \phi$. The degree of a separable isogeny is given by its degree as a rational map and the size of its kernel. An isogeny is *cyclic* if its kernel is a cyclic group. For every isogeny $\phi : E \to E'$, there exists a unique dual isogeny $\hat{\phi} : E' \to E$ such that $\hat{\phi} \circ \phi = [\deg \phi]$, where $[m]$ denotes the multiplication-by-m map of a curve. Given there exists an isogeny $\phi : E \to E'$ of degree N, we say that E and E' are N-isogenous, and that ϕ is an N-isogeny. In this work, we assume all isogenies are separable and cyclic (unless otherwise stated).

Over a finite field $\mathbb{F}_{p^2}$, the endomorphism ring of elliptic curves either form (i) an imaginary quadratic field, in which case we say the curve is *ordinary*, or (ii) a maximal order in a quaternion algebra, in which we say the curve is *supersingular.* The *ℓ-supersingular isogeny graph over* $\mathbb{F}_{p^2}$, is a graph constructed by taking the vertices to be the set of supersingular elliptic curves over $\mathbb{F}_{p^2}$ up to isomorphism (often labeled by their j-invariant), and the edges to be all ℓ-isogenies between curves. It is a well known fact that the graph is connected and $\ell + 1$-regular for all primes ℓ. First introduced by Pizer [64], these graphs are *Ramanujan*, which means that the distribution of random walks taken on the graph quickly approaches the uniform distribution on the vertex set. The conventional isogeny-based hardness assumption is defined below.

Problem 1 (ℓ^n-isogeny path problem). Given two supersingular elliptic curves E, E' over $\mathbb{F}_{p^2}$, find a cyclic isogeny $\phi : E \to E'$ of degree $\deg \phi = \ell^n$.

The hardness of the problem above allows the constructions of one way functions, given a way to map inputs to unique isogenies, such as the CGL hash function and subsequent works [23, 38].

The CGL hash function. The first isogeny-based protocol was the CGL hash function, introduced by Charles, Goren and Lauter in [23]. The function, given a public supersingular starting curve E over $\mathbb{F}_{p^2}$, takes a non-backtracking walk in the supersingular ℓ-isogeny graph, dictated by some input k. This can be done by parsing the input string in base ℓ, e.g. $k = k_0k_1 \dots k_n$ with $k_i \in \{0, \dots, \ell - 1\}$ and ordering the outgoing (non-backtracking) edges from every vertex E_i in some canonical way, then choosing the path that corresponds to the digit k_i. After n steps, one arrives at the output curve E_n. The CGL hash function is provably preimage resistant given the hardness of Problem 1, and collision resistant provided either the endomorphism ring of the starting curve is unknown, or the input length is fixed and small enough to prevent endomorphism ring attacks. A convenient way of instantiating the CGL hash function is by using the supersingular 2-isogeny graph, which is 3-regular. Thus, by preventing backtracking, the direction of each step of the walk is determined by a single bit of the input.

2.2 Effective group actions

When working with elliptic curves and isogenies over a prime field $\mathbb{F}_p$, isogeny computations can be abstracted using the group action framework. For an introduction into group actions from isogenies, we point the interested reader to the original paper by Couveignes [28] as well as to the CSIDH paper [22] for its instantiation in the supersingular case. For more details, we also recommend the introductory notes by De Feo [30]. In this section, we satisfy ourselves with introducing the group action framework abstractly. In particular, throughout this work, we will only consider *effective group actions* (EGAs) as introduced in [1].

Definition 1 (Effective group action [1]). *A group $(G, +)$ is said to* act *on a set $\mathcal{E}$, if there exists a map $[\,] : G \times \mathcal{E} \to \mathcal{E}$, which satisfies the following two properties:*

- *Identity: if 0 is the identity element of G, then for any $E \in \mathcal{E}$, we have $[0]E = E$.*
- *Compatibility: For any $a, b \in G$ and any $E \in \mathcal{E}$, we have $[a+b]E = [a]([b]E)$.*

A group action is said to be effective*, if*

- *The group G is finite and there exist efficient algorithms for sampling, membership testing, equality testing and inversion of group elements, as well as for performing the group operation between group elements.*
- *The set $\mathcal{E}$ is finite and there exist efficient algorithms for membership testing. In addition, every element $E \in \mathcal{E}$ has a unique representation.*
- *The group action computation is efficient.*
- *There exists a distinguished element E_0 of known representation, called the* origin.

An example of an EGA from isogenies between supersingular elliptic curves has first been computed in [13] and can also be realized using the approaches from [25, 32] or from [60]. It is important to note that in general, the group G is isomorphic to $\mathbb{Z}_N := \mathbb{Z}/N\mathbb{Z}$, where N is a composite number.[5] EGAs from isogenies generally enjoy the following property via the twisting operation.

Definition 2 (Symmetric EGA [6]). *An effective group action is called* symmetric *around the distinguished element E_0 if there exists an efficient algorithm (called* twisting*), that given $E = [a]E_0$ computes $E^t = [-a]E_0$ without any extra information.*

We also note that effective group actions from isogenies are both *commutative* and *regular*. For completeness, we define the latter below.

Definition 3 (Regularity [1]). *A (effective) group action is called* regular *if it satisfies the following two properties.*

- *Free: no non-trivial element of G fixes an element of $\mathcal{E}$, i.e. if $[a]E = E$ for some $E \in \mathcal{E}$, then $a = 0$.*
- *Transitive: for every $E, E' \in \mathcal{E}$, there exists some $a \in G$, s.t. $E' = [a]E$.*

We note that regularity immediately implies that $|G| = |\mathcal{E}|$, since for any $E \in \mathcal{E}$, the map $a \mapsto [a]E$ defines a bijection between G and $\mathcal{E}$.

Cryptographic group actions generally come with some computational hardness assumptions. Two of the most important problems, that we will assume to be hard throughout this work, are the following.

Problem 2 (Key recovery). Given $E, E' \in \mathcal{E}$, find $a \in G$, such that $E' = [a]E$.

Problem 3 (Computational Diffie-Hellman). Given $E, E', F \in \mathcal{E}$, where $E' = [a]E$, compute F' such that $F' = [a]F$.

[5] We would like to note that G describes the exponent group of a generator $\mathfrak{g}$ (i.e. $[a]E$ denotes the isogeny of kernel $\{P \in E \mid \alpha(P) = O \text{ for all } \alpha \in \mathfrak{g}^a\}$.) of the ideal-class group of orders in quadratic imaginary fields, $\mathsf{cl}(\mathcal{O})$, which we assume to be cyclic (or working in a cyclic subgroup). Class groups of this type generally have non-prime order $N = \#\mathsf{cl}(\mathcal{O})$.

2.3 Verifiable random functions

Verifiable random functions were first formalized by Micali, Rabin and Vadhan [57]. The authors define a verifiable random function (VRF) as a tuple of polynomial time algorithms $VRF = (\mathsf{SetUp}, \mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Verif})$ with the properties below. Let $\mathcal{K}$ be the secret key space, $\mathcal{M}$ the input/message space.

- $\mathsf{SetUp}(1^\lambda)$ takes as input a security parameter λ and returns public parameters pp.
- $\mathsf{KeyGen}(\mathsf{pp})$ takes as input public parameters pp and returns a secret-public key pair $(\mathsf{sk}, \mathsf{pk})$.
- $\mathsf{Eval}_{\mathsf{sk}}(m)$ takes as input a secret key sk and an input string $m \in \mathcal{M}$, then returns an output value h and a proof π of the correctness of h.
- $\mathsf{Verify}_{\mathsf{pk}}(m, h, \pi)$ takes as input the public key pk, the output h and proof π, as well as the input m and returns either 1 or 0, indicating that it accepts or rejects the proof.

The security of VRFs is formalized through three security properties. *Provability* states that any correct evaluation of the VRF should result in an output pair that passes the verification algorithm. *Unique provability* further implies that this output pair is unique, i.e. that for a given input and public key, there do not exist distinct outputs that correctly verify. Finally, any adversary interacting with the VRF-functionality should not be able to find an input-output pair that passes verification. This is formalized in the *residual pseudorandomness* property.

Definition 4 (Provability). *For any input $m \in \mathcal{M}$, a correctly generated evaluation will result in an accepting proof with overwhelming probability. Formally, the following holds:*

$$\Pr\left[\mathsf{Verify}_{\mathsf{pk}}(m, h, \pi) = 1 \;\middle|\; \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{SetUp}(1^\lambda) \\ (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ (h, \pi) \leftarrow \mathsf{Eval}_{\mathsf{sk}}(m) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda)\,.$$

Definition 5 (Unique Provability). *For any public key* pk *and any input m, there does not exist two distinct evaluations which both have accepting proofs, except with negligible probability. Formally, for all adversaries $\mathcal{A}$, which may be computationally unbounded (with at most polynomially many public coin queries),*

$$\Pr\left[\begin{array}{c} \mathsf{Verify}_{\mathsf{pk}}(m, h_1, \pi_1) = 1 \,\wedge \\ \mathsf{Verify}_{\mathsf{pk}}(m, h_2, \pi_2) = 1 \,\wedge\; h_1 \neq h_2 \end{array} \;\middle|\; \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{SetUp}(1^\lambda) \\ (\mathsf{pk}, m, h_1, \pi_1, h_2, \pi_2) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp}) \end{array}\right]$$

is negligible in the security parameter λ.

Definition 6 (Residual Pseudorandomness). *Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a PPT adversary and H be a random oracle in the following game:*

1. $\mathsf{pp} \leftarrow \mathsf{SetUp}(1^\lambda)$
2. $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$

3. $(m^*, st) \leftarrow \mathcal{A}_1^{\mathsf{Eval}_{\mathsf{sk}}(\cdot), H(\cdot)}(\mathsf{pk})$
4. $(h_0, \pi_0) \leftarrow \mathsf{Eval}_{\mathsf{sk}}(m^*)$
5. $h_1 \leftarrow \{0,1\}^{2\lambda}$
6. $b \leftarrow \{0,1\}$
7. $b' \leftarrow \mathcal{A}_2^{\mathsf{Eval}_{\mathsf{sk}}(\cdot), H(\cdot)}(\mathsf{pk}, h_b, st)$

$\mathcal{A}$ wins, if $b = b'$ and if it hasn't queried $\mathsf{Eval}_{\mathsf{sk}}(m^*)$*. A VRF satisfies the* residual pseudorandomness *property, if $Pr[\mathcal{A}\ wins] \leq \mathsf{negl}(\lambda)$.*

In this work, we also consider the following new relaxed notion of unique provability, which we call *weak unique provability.* In some cases it is impractical to ensure that the underlying key generation function in a VRF is injective. If this is not the case, an unbounded adversary for Definition 5 can search for a collision $\mathsf{sk}, \mathsf{sk}'$ such that $\mathsf{Verify}_{\mathsf{pk}}(\mathsf{Eval}_{\mathsf{sk}}(m)) = \mathsf{Verify}_{\mathsf{pk}}(\mathsf{Eval}_{\mathsf{sk}'}(m))$, where in general it is not the case that $\mathsf{Eval}_{\mathsf{sk}}(m) = \mathsf{Eval}_{\mathsf{sk}'}(m)$. This relaxed definition accounts for the non-uniqueness of secret keys corresponding to a fixed public key. Rather, that it guarantees that finding colliding secret keys are computationally hard to find by requiring that the adversary is polynomial time bounded in the security parameter.

Definition 7 (Weak Unique Provability). *It is computationally infeasible to construct a public key* pk *and input m such that there exists two distinct evaluations which both have accepting proofs. Formally, for all PPT adversaries $\mathcal{A}$, the probability*

$$\mathsf{Pr}\left[\begin{array}{c}\mathsf{Verify}_{\mathsf{pk}}(m, h_1, \pi_1) = 1\ \wedge \\ \mathsf{Verify}_{\mathsf{pk}}(m, h_2, \pi_2) = 1\ \wedge\ h_1 \neq h_2\end{array} \middle| \begin{array}{c}\mathsf{pp} \leftarrow \mathsf{SetUp}(1^\lambda) \\ (\mathsf{pk}, m, h_1, \pi_1, h_2, \pi_2) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp})\end{array}\right]$$

is negligible in the security parameter λ.

2.4 Zero-knowledge proof systems

Let $\mathcal{R} : X \times W \to \{0,1\}$ be a relation with input set X and witness set W defining the NP-language $\mathcal{L} = \{x \in X : \exists w \in W \text{ s.t. } (x, w) \in \mathcal{R}\}$. Zero-knowledge proofs are protocols between two parties, where a prover P tries to convince a verifier V that, given some $x \in X$, that $x \in \mathcal{L}$, i.e. that there exists (or it knows) a witness $w \in W$ such that $(x, w) \in \mathcal{R}$ [48].

Zero-knowledge proofs may be interactive protocols, but can generally be made non-interactive in the Random Oracle Model [9]. We are primarily interested in non-interactive zero-knowledge *proofs of knowledge* in this work and will, on occasion, abbreviate them as either NIZK or NIZKPoK. We further write H for the random oracle. We outline the formal definitions of the security properties of a NIZKPoK below.

Definition 8 (Completeness). *For every $(x, w) \in \mathcal{R}$, honest prover P and honest verifier V, it holds that*

$$\Pr[V^{\mathsf{H}}(x, \pi) = 1 \mid \pi \leftarrow P^{\mathsf{H}}(x, w)] = 1$$

Definition 9 (Soundness). *For any $x \notin \mathcal{L}$, PPT prover P and honest verifier V, it holds that*

$$\Pr[V^{\mathsf{H}}(x, \pi) = 1 \mid \pi \leftarrow P^{\mathsf{H}}(x)] \leq \mathsf{negl}(\lambda)$$

Definition 10 (Knowledge Soundness). *We say that a protocol is knowledge sound, with knowledge error κ, if there exists a* PPT *extractor E such that, for every x,* PPT *$\tilde{P}$, $\lambda \in \mathbb{N}$,*

$$\Pr[(x, w) \in \mathcal{R} \mid w \leftarrow E^{\tilde{P}}(x, 1^{\lambda})] \geq \Pr[V^{H}(x, \pi) = 1 \mid \pi \leftarrow \tilde{P}^{H}] - \kappa(x, \lambda).$$

Where the extractor E may program the responses to random oracle queries of $\tilde{P}$, and either get a response of the next query or output π, at which point $\tilde{P}$ goes to the start of its computation with the same randomness and auxiliary input.

Definition 11 (Zero Knowledge). *A non-interactive protocol (P, V) is (computational/statistical/perfect) zero-knowledge (with negligible function z) in the random oracle model, if there exists a* PPT *simulator $\mathcal{S}$, such that for every $(x, w) \in \mathcal{R}$, the following distributions:*

$$\{\pi \leftarrow \mathcal{S}^{H}(x)\}, \textit{ and } \{\pi \leftarrow P^{H}(x, w)\},$$

are (computationally/statistically/perfectly) indistinguishable, where the distributions are taken over the uniformly random instantiation of H and the randomness of P.

In this work, we construct a NIZKPoK by means of a sigma-protocols, which is a 3-round public coin interactive proof performed in the following phases on common input x:

Commitment: A prover sends a message comm to the verifier.
Challenge: On receiving comm sends a random $\mathsf{chall} \leftarrow\!\!\$\; \mathcal{C}$ from a challenge space $\mathcal{C}$.
Response: on receiving chall, sends a response resp.
Verification: After interaction, on input $(x, \mathsf{comm}, \mathsf{chall}, \mathsf{resp})$ the verifier outputs either 1 (accept) or 0 (reject).

which satisfy completeness, t-special soundness, and honest verifier zero knowledge.

Definition 12 (Honest Verifier Zero-Knowledge (HVZK)). *There exists a PPT simulator $\mathcal{S}$ such that on input (x, chall), outputs valid transcripts*

$$(\mathsf{comm}, \mathsf{resp}) \leftarrow \mathcal{S}(x, \mathsf{chall})$$

in a distribution that is (computationally/statistically/perfectly) indistinguishable from the distribution of real transcript from an honest prover with input (x, w) and challenge chall.

Definition 13 **(t-special soundness).** *There exists a PPT algorithm E called the extractor, which given instance x, and t valid distinct transcripts*

$$[(\mathsf{comm}_i, \mathsf{chall}_i, \mathsf{resp}_i)_{i\in[t]}]$$

where $\mathsf{comm}_i = \mathsf{comm}_j$ *(with a common first message),* $\mathsf{chall}_i \neq \mathsf{chall}_j$ *for all* $1 \leq i < j \leq t$*,* E *outputs* w *such that* $(x, w) \in \mathcal{R}$*.*

When the Fiat-Shamir [43] transform is applied to a Sigma-protocol which is complete, t-special sound, and honest verifier zero-knowledge, the resulting protocol is a NIZKPoK with zero-knowledge, and knowledge error $\frac{t-1}{c}$ where c is the size of the verifier's challenge space.

Finally a *zero-knowledge succinct non-interactive argument of knowledge* (zk-SNARK) is a NIZKPoK, which in addition to satisfying completeness, knowledge soundness and zero-knowledge, also satisfies *succinctness*, which is that proofs scale polylogarithmically in the size of the witness. A core idea underlying current zk-SNARK constructions is to encode the proof into a circuit with particular constraints reflecting the constraints of the statement, and then to show that indeed the circuit is satisfied by the statement of the prover. There are many different ways to encode such circuits. In this work, we will focus on Rank-1 Constraint Systems (R1CS), which we recall here.

Let $\mathbb{F}_q$ be a finite field and let $m, n \in \mathbb{N}$ be parameters. R1CS aims to encode statements into the form

$$Az \circ Bz = Cz\,,$$

where $A, B, C \in \mathbb{F}_q^{m\times n}$ are matrices, $z \in \mathbb{F}_q^n$ a vector and $\circ$ is the Hadamard product. Conceptually, the matrices A, B, C encode constraints on the variables in $z := (1, v, w)$, where v is part of the input and contains auxiliary public input, while w contains both the secret input of the prover and intermediate variables in a computation. The types of relations of R1CS proof systems are to show are therefore statements of the type $(A, B, C, v) \in \mathcal{L}$ using the witness w, i.e.

$$((A, B, C, v), w) \in \mathcal{R}_{R1CS}\,.$$

3 Faster proofs of isogeny walks

In this section, we develop new R1CS descriptions for proofs of isogeny knowledge in the 2-isogeny graph. We provide both an optimised R1CS description for proving knowledge of a 2^n-isogeny for any n as in [27], and a 2-isogeny walk predetermined by a directional bitstring, essentially that a CGL instance has been computed correctly. We extend the latter to allow the proof that two different paths have been computed using the same string as an input.

We start this section with a radical isogeny version of CGL, which is interesting in its own right. We then move on to building a proof system of correct CGL evaluation, and finally simplify/extend this to the above-mentioned proofs.

3.1 Radical isogeny CGL

In the 2-isogeny graph, every vertex has three outgoing isogenies, so at every step except the first, without backtracking, we have exactly two options to continue our path. In the original CGL construction, the 2-torsion is ordered in some deterministic fashion and the input k, read as bits, determines whether to go "left" ($k_i = 0$) or "right" ($k_i = 1$) at every bifurcation.

We modify this construction by using the *radical isogeny* formulas from [21] (we note that the formula for $\ell = 2$ have actually already been proposed in [20]). In their work, the authors introduce a faster way to compute 2-isogenies, which removes the need to generate the 2-torsion in the first place, thus speeding up the computation. The dominant part of the isogeny computation then becomes taking a square-root. An isogeny of degree 2 connecting two curves E_i and E_{i+1} can then be expressed as

$$E_i : y^2 = x^3 + A_i x^2 + C_i x \quad \longrightarrow \quad E_{i+1} : y^2 = x^3 + A_{i+1} x^2 + C_{i+1} x \,,$$

where the coefficients are connected via the following simple formulas

$$A_{i+1} = 6\sqrt{C_i} + A_i \,, \quad C_{i+1} = 4\sqrt{C_i} A_i + 8C_i \,.$$

Clearly, there are two choices for $\sqrt{C_i}$. In fact, in the 2-isogeny graph, this choice corresponds to going left or right in a non-backtracking walk, in a similar way as CGL does. There are many ways to distinguish these two roots, but since we plan to later distinguish them algebraically as part of our R1CS constraints, we use the following method.

First, let us describe $\mathbb{F}_{p^2} = \mathbb{F}_p(j)$ with $j^2 = d$ a quadratic non-residue. For an element $a + bj \in \mathbb{F}_p(j)$, we write $\mathrm{Re}(a+bj) = a$ and $\mathrm{Im}(a+bj) = b$. Now, let $\pm\alpha_i$ be the square roots of C_i. By working over $p \equiv 3 \pmod 4$, we can distinguish $\pm\alpha_i$ via the residuosity of their real (or imaginary) part: since $\mathrm{Re}(\alpha_i) \in \mathbb{F}_p$, we find that if $\mathrm{Re}(\alpha_i)$ is a quadratic residue and $\mathrm{Re}(-\alpha_i)$ is not and vice versa.

From here, we define α_i as the square root of C_i for which $\mathrm{Re}(\alpha_i)$ is itself a square. We reinterpret the bits of an input k as $k_i \in \{-1, 1\}$. If $k_i = 1$, we walk the path in our bifurcation that is determined by $+\alpha_i$, and if $k_i = -1$, we walk the path determined by $-\alpha_i$. We can rewrite the relation between the curve coefficients as

$$A_{i+1} = 6k_i\alpha_i + A_i \,, \quad C_{i+1} = 4k_i\alpha_i A_i + 8C_i \,.$$

Similarly to CGL, this allows us to feed in an input $k = k_0 k_1 \dots k_{n-1}$, that results in a deterministic path from a starting curve E_0 to some output E_n. We note that the choice of parameters of the starting curve immediately excludes one possible path using the formulas mentioned above, which correspond to the isogeny generated by the kernel point $(0, 0)$, so we do not need to take extra care at this first step. Furthermore, backtracking is excluded by the nature of the formulas themselves. We summarize our algorithm in Figure 1 below.

```
Input: Supersingular elliptic curve $E_0$, message $k = k_0 k_1 \dots k_{n-1} \in \{-1, 1\}^n$
Output: Supersingular elliptic curve $E_n$.

    1. Let $E_0 : y^2 = x^3 + A_0 x^2 + C_0 x$. For $i = 0, \dots, n-1$, do the following
        (a) Let $\alpha_i$ be the square root of $C_i$, such that $\mathrm{Re}(\alpha_i)$ is a square.
        (b) Compute $A_{i+1} = 6k_i\alpha_i + A_i$ and $C_{i+1} = 4k_i\alpha_i A_i + 8C_i$.
    2. Return $E_n$.
```

Fig. 1: Radical isogeny CGL

3.2 The proof system

We are now building a proof system for the following relation

$$\mathcal{R}_{CGL} = \{(E_0, E_n), k : E_n = \mathsf{CGL}(E_0, k)\} ,$$

where we denote $\mathsf{CGL}(E_0, k)$ as the correct output of the algorithm in Figure 1 with input some starting curve E_0 and a secret input $k = k_0 k_1 \dots k_n \in \{-1, 1\}^n$. The proof system needs to show that steps 1(a) and 1(b) have been computed correctly, using the correct k_i at every step.

We describe our proof system as a Rank-1 Constraint System (R1CS). The easiest way to prove that $\mathrm{Re}(\alpha_i)$ is indeed a square is to again provide the square root, which we will denote as β_i, an element of $\mathbb{F}_p$. For β_i, we do not need to distinguish which root we are talking about, only that $\mathrm{Re}(\alpha_i)$ has a root. Together with the equations from 1(b), we find the following system of equations

$$\beta_i^2 = \mathrm{Re}(\alpha_i) \tag{1}$$

$$\alpha_i^2 = C_i \tag{2}$$

$$A_{i+1} - A_i = 6k_i\alpha_i \tag{3}$$

$$C_{i+1} - 8C_i = 4k_i\alpha_i A_i \, . \tag{4}$$

By plugging in (3) into (4) and scaling, we can reduce the triple term in (4) to a simple multiplication

$$3C_{i+1} - 24C_i = 2A_i(A_{i+1} - A_i) ,$$

which is much more convenient for R1CS, as it results in less constraints.

Since we want to distinguish square roots via their residuosity, we need to reinterpret $\mathbb{F}_{p^2} = \mathbb{F}_p(j)$ as $\mathbb{F}_p \times \mathbb{F}_p$. We can write the square equation (2) over $\mathbb{F}_{p^2}$ as two constraints over $\mathbb{F}_p \times \mathbb{F}_p$, while the multiplication in equation (4) can be written as three constraints, using an auxiliary value (see e.g. [27, Section 3.5]). Note that the multiplication in (3) is a multiplication of a scalar with an element from $\mathbb{F}_p \times \mathbb{F}_p$, thus also resulting in two constraints. Finally, we have to ensure is that all elements k_i are indeed in $\{-1, 1\}$, which can be done by showing that $k_i^2 = 1$. We defer the full constraints including input-output description and matrices to Appendix A. We note here that for a path of length n, the constraint system amounts to

$$9n + 4 \text{ variables and } 9n \text{ constraints.}$$

3.2.1 Proof of isogeny knowledge. In case we simply want to prove knowledge of an isogeny path of length n in the 2-isogeny graph (without caring about the input value k), i.e. an instance of the relation

$$\mathcal{R}_{2^n\text{-iso}} = \{(E_0, E_n), \phi \ : \ \phi : E_0 \to E_n \text{ with } \deg \phi = 2^n\} \ ,$$

we can simplify our R1CS description introduced in the previous section. We can do this by simply dropping the constraint expressed by equation (1), as we don't need to prove anything about the residuosity of $\mathrm{Re}(\alpha_i)$, only that indeed $\alpha_i^2 = C_i$ at every step. This also simplifies equations (3) and (4) to

$$A_{i+1} - A_i = 6\alpha_i \quad \text{ and } \quad C_{i+1} - 8C_i = 4\alpha_i A_i \, .$$

Finally, we don't need to prove that $k_i^2 = 1$, so that in the end, over $\mathbb{F}_p \times \mathbb{F}_p$, we are left with

$$7n + 5 \text{ variables and } 7n \text{ constraints.}$$

At this point, we would like to compare our description to [27], in which the authors first introduced R1CS as a tool to prove knowledge of an isogeny. Rather than using radical isogeny formulas, the authors based their proof system on expressions using modular polynomials, which include quadratic and cubic terms in the j-invariants of elliptic curves. For a path of length n, the authors express the constraint system over $\mathbb{F}_p \times \mathbb{F}_p$ using

$$12n + 4 \text{ variables and } 12n + 3 \text{ constraints,}$$

including the non-backtracking condition from their Appendix A (which we get for free using radical isogeny formulas). It is easy to see that our results outperform theirs at no extra cost. We note however that our proof system works only for $p \equiv 3 \pmod 4$ (which is standard for isogeny-based cryptography), while the results from [27] do not have this restriction.

3.2.2 Proof of same input. For some applications (hinting towards Section 4), one might want to prove that two paths in the 2-isogeny graph have been computed using the same input k, starting from different elliptic curves. Let for example (E_1, E_2) and (F_1, F_2) be two tuples of elliptic curves, then we can define the corresponding relation as

$$\mathcal{R}_{\mathsf{CGL}/\!\!/} = \{(E_1, E_2), (F_1, F_2), k \ : \ E_2 = \mathsf{CGL}(E_1, k) \wedge F_2 = \mathsf{CGL}(F_1, k)\} \ . \quad (5)$$

This implies that we need a constraint system that realizes equations (1)-(4) twice in parallel, with the same k as input, and show again that $k_i^2 = 1$ at every step. In total, this simple modification results in a proof system with

$$17n + 8 \text{ variables and } 17n \text{ constraints}$$

over $\mathbb{F}_p \times \mathbb{F}_p$. We denote running this proof as $\pi \leftarrow \mathsf{NIZK}^{/\!/}.P(k, (E_1, E_2, F_1, F_2))$, while verifying it as $0/1 \leftarrow \mathsf{NIZK}^{/\!/}.V(\pi, (E_1, E_2, F_1, F_2))$.

4 A new verifiable random function from isogeny walks

In this section we present a new construction for a verifiable random function (VRF). In order to stay general, we start with a construction from what we will call *unpredictable functions*, a related notion to the functions introduced in the original work of [57]; together with general non-interactive zero knowledge arguments, and prove its security. We show later that CGL walks along with our proof systems from the previous section are a plausible instantiation of this VRF construction, but we also believe that instantiations from other paradigms might be possible, and leave this open for further research.

4.1 A generic VRF construction

We will use the following syntax throughout this section.

Definition 14 (Unpredictable function). *Let $\mathcal{E}$ and $\mathcal{K}$ be sets, and let*

$$\mathcal{W} : \mathcal{E} \times \mathcal{K} \to \mathcal{E}$$

be a deterministic function. We further define $\mathsf{SetUpUF}(1^\lambda)$ *as a function which on input a security parameter λ outputs an unpredictable function $(\mathcal{W}, \mathcal{E}, \mathcal{K})$ together with an element $E_0 \in \mathcal{E}$ called the* starting element. *We call $(\mathcal{W}, \mathcal{E}, \mathcal{K})$* unpredictable, *if the one-more evaluation problem on $\mathcal{W}$ is hard.*

Problem 4 (One-more evaluation problem). Let $\mathcal{O}_k(\cdot)$ be an evaluation oracle, which for given public parameters pp and on input $m \in \mathcal{K}$ returns $E \leftarrow \mathcal{W}(\mathcal{W}(E_0, m), k)$. Finally, let $\mathcal{A}$ be a PPT adversary for which we define the following game. On input λ:

1. $\mathsf{pp} \leftarrow \mathsf{SetUpUF}(1^\lambda)$. Parse pp as $(\mathcal{W}, \mathcal{E}, \mathcal{K}, E_0)$.
2. $k \leftarrow\!\!\$\ \mathcal{K}$
3. $E_k \leftarrow \mathcal{W}(E_0, k)$
4. $(m^*, E^*) \leftarrow \mathcal{A}^{\mathcal{O}_k(\cdot)}(E_k, \mathsf{pp})$
5. $\mathcal{A}$ wins if $E^* = \mathcal{W}(\mathcal{W}(E_0, m^*), k)$ and m^* has not been queried to $\mathcal{O}_k(\cdot)$ before, and loses otherwise.

Note that the assumption that Problem 4 is hard implies that unpredictable functions $\mathcal{W}$ are collision-resistant, since finding m, m^* where $\mathcal{W}(E_0, m) = \mathcal{W}(E_0, m^*)$ would obviously break Problem 4. Furthermore, we have that $\mathcal{W}$ is non-commutative in the following sense

$$\mathcal{W}(\mathcal{W}(E_0, m), k) \neq \mathcal{W}(\mathcal{W}(E_0, k), m)\,.$$

Otherwise, one could trivially break the assumption by sampling m^* and computing $\mathcal{W}(E_k, m^*)$.

Proof system. We let $\pi \leftarrow \mathsf{NIZK}.P(k, (E_1, E_2, F_1, F_2); \mathsf{pp})$ designate a non-interactive zero-knowledge proof, which for a given input value k and tuples (E_1, E_2) and (F_1, F_2), outputs a proof π for the following relation.

$$\mathcal{R} = \Big\{(E_1, E_2), (F_1, F_2), k \;:\; E_2 = \mathcal{W}(E_1, k) \wedge F_2 = \mathcal{W}(F_1, k)\Big\}\,. \tag{6}$$

A verifier can then run $0/1 \leftarrow \mathsf{NIZK}.V(\pi, (E_1, E_2, F_1, F_2); \mathsf{pp})$ in order to verify a proof π with regards to two tuples (E_1, E_2) and (F_1, F_2). The verifier outputs 0, if it rejects the proof, and 1 otherwise.

4.1.1 Constructing VRFs from unpredictable functions.

In Figure 2, we present our construction for a VRF based on an unpredictable function $\mathcal{W}$ and the proof system NIZK. The VRF evaluation simply consists of two consecutive evalutions of $\mathcal{W}$, first with input m, then with input k, and a proof that the second path was computed correctly, i.e. using the secret key k that defines the public key E_k.

$$\begin{aligned} E_0 \xrightarrow{\quad m \quad} E_m = \mathcal{W}(E_0, m) \xrightarrow{\quad k \quad} E = \mathcal{W}(E_m, k) \\ \text{and,} \qquad E_0 \xrightarrow{\; k \;} E_k = \mathcal{W}(E_0, k) \end{aligned} \tag{7}$$

At the end, the output, together with the input and the public key, are hashed using a random oracle $H : \{0,1\}^* \to \{0,1\}^{2\lambda}$. This allows to base our underlying hardness assumption on the computational hardness of Problem 4, rather than a decisional one. We prove security of our construction in Appendix B.1, and state the overall result in the following theorem:

Theorem 1. *Let NIZK be a NIZKPoK for the relation 6 and let $\mathcal{W}$ be an injective unpredictable function. Then Figure 2 is a verifiable pseudorandom function (VRF) in the random oracle model. If $\mathcal{W}$ is not injective (but still collision resistant), then the protocol is a verifiable pseudorandom function (VRF) with weak unique provability.*

Remark 2. We note that lifting to a VRF via unpredictable functions is considered folklore in the literature, but its security was not given formal treatment (in

```
**SetUp**$(1^\lambda)$:
    1. Return $\mathsf{pp} = (\mathcal{W}, \mathcal{E}, \mathcal{K}, E_0) \leftarrow \mathsf{SetUpUF}(1^\lambda)$.
**KeyGen**$(\mathsf{pp})$:
    1. Sample $k \leftarrow \mathcal{K}$.
    2. Compute $E_k = \mathcal{W}(E_0, k)$.
    3. Return $(\mathsf{sk}, \mathsf{pk}) = (k, E_k)$
**Eval**$_k(m; \mathsf{pp})$:
    1. Verify that $m \in \mathcal{K}$.
    2. Compute $E_m = \mathcal{W}(E_0, m)$
    3. Compute $E = \mathcal{W}(E_m, k)$
    4. Run $\pi_1 \leftarrow \mathsf{NIZK}.P(k, (E_0, E_k, E_m, E); \mathsf{pp})$.
    5. Return the output $h = H(E_k, m, E)$ and the proof $\pi = (\pi_1, E)$.
**Verify**$_{\mathsf{pk}}(h, (\pi_1, E), m; \mathsf{pp})$:
    1. Compute $E_m = \mathcal{W}(E_0, m)$.
    2. Return $h \overset{?}{=} H(E_k, m, E) \wedge \mathsf{NIZK}.V(\pi_1, (E_0, E_k, E_m, E); \mathsf{pp})$.
```

Fig. 2: Verifiable pseudorandom function from unpredictable function $\mathcal{W}$ and zero-knowledge proof NIZK.

the random oracle model in particular) until recently. During the writing of this work, a concurrent result [46] also proved the security of this construction in the random oracle model, using a closely related notion of *Verifiable Unpredictable Functions* (VUFs). VUFs are unpredictable functions with the verifiability constraint included in their definition. Given that unpredictable functions equipped with an NIZKPoK for relation (6) satisfy the property of VUF, their result implies that our resulting VRF construction, following the compiler from [46, Fig. 9], also satisfies the stronger security notion of *unbiasability*. This property required to guarantee fairness in VRF-based leader election, as required in VRF-based proof-of-stake protocols. We refer the reader to [46] for more details.

4.2 Instantiating our VRF via the CGL Hash

In this section, we discuss why isogenies are well-suited candidates to instantiate our VRF construction from Figure 2. In particular, we instantiate the unpredictable function $\mathcal{W}$ using the radical isogeny protocol CGL from Figure 1 over supersingular elliptic curves. Formally, we define $\mathsf{SetUpUF}(1^\lambda)$ as returning $(\mathsf{CGL}, \mathcal{E}, \mathcal{K}, E_0)$, where

- CGL is the function described in Figure 1,
- $\mathcal{E}$ defines the set of supersingular elliptic curves over a finite field $\mathbb{F}_{p^2}$ with parameter size defined with respect to the security parameter λ (we discuss actual parameters in 4.2.1),
- $E_0 \in \mathcal{E}$ is a starting curve, and

– $\mathcal{K}$ is the set of input strings $\{-1,1\}^n$.

We can then interpret the VRF input $m \in \{-1,1\}^n$ as a fixed-length, binary string which defines a walk from the starting curve E_0 to some curve E_m, from which we then start another walk, this time defined by the server's secret key $k \in \{-1,1\}^n$ towards the final curve E. For the proof system, we use the proof of same input $\mathsf{NIZK}^{/\!/}$ described in Section 3.2.2. The idea is for the server to show that it used its secret key k, which connects E_0 to the public key $E_k = \mathsf{CGL}(E_0,k)$, also as an input to evaluate the VRF. We get the picture below

$$\begin{aligned} E_0 &\xrightarrow{\quad m \quad} E_m = \mathsf{CGL}(E_0,m) \xrightarrow{\quad k \quad} E = \mathsf{CGL}(E_m,k) \\ &\text{and,} \qquad E_0 \xrightarrow{\ k\ } E_k = \mathsf{CGL}(E_0,k)\,, \end{aligned} \tag{8}$$

where the $\mathsf{NIZK}^{/\!/}$ proves that indeed the pairs (E_m,E) and (E_0,E_k) are connected by the same k, according to relation (5). In the next section, we discuss the suitability of CGL as an unpredictable function and motivate the security of our instantiation.

4.2.1 Security considerations. We first discuss considerations which need to be taken into account when implementing the CGL hash function in order to guarantee collision and pre-image resistance, which are necessary to achieve unpredictability. We then motivate the hardness of the one-more evaluation problem and discuss secure parameter sizes and efficiency of our protocol.

Non-backtracking walks. We first note that the walks on the isogeny graph must be non-backtracking, otherwise pre-image and collision resistance can be trivially broken. By using the radical isogeny formulas, this is automatically avoided, as these are non-backtracking by design [21]. Each iteration only allows for two choices of outgoing isogenies, never the dual of the prior step.

Pre-image resistance. Given that the walks are non-backtracking, for a curve E_m and a valid output curve E_k, computing a pre-image k such that $E_k = \mathsf{CGL}(E_m,k)$ corresponds to computing a cyclic 2^e-isogeny from E_m to E_k in the isogeny graph, which is Problem 1. The best known attacks on this problem run in time $O(2^{e/2})$ via claw-finding attacks [34] and $\tilde{O}(\sqrt{p})$ via the volcano-finding algorithm of [45]. Hence, we require $e \geq 2\lambda$ and $\log p \geq 2\lambda$.

Collision resistance and endomorphism ring attacks. The function $\mathsf{CGL}(E_0,\cdot)$ defined over a fixed input length e is not in general injective Furthermore, in case the attacker has knowledge of the endomorphism ring of the starting curve, they can compute collisions as described in the attacks of [63, Section 4.2], by using the KLPT algorithm [53]. In particular, in order to compute collisions, the attacker computes cycles in the 2-isogeny graph from the starting curve. This attack can be prevented in two ways, either (1) by using a starting curve E_0

with unknown endomorphism ring, via a trusted setup ceremony via techniques described in [8], or (2) by limiting the message space to be short enough such there are no endomorphisms of length 2^r for $r \leq 2n$. Existence of such cycles would admit a collision in the function $\mathsf{CGL}(E_0, \cdot)$, leading to the scenario where $\mathsf{pk} = \mathsf{CGL}(E_0, \mathsf{sk}) = \mathsf{CGL}(E_0, \mathsf{sk}')$ for distinct $\mathsf{sk}, \mathsf{sk}'$, and $\mathsf{Eval}_{sk}(m) \neq \mathsf{Eval}_{sk'}(m)$, violating weak unique provability, and a cycle of length 2^{2n} would allow an adversary to find distinct m, m' such that $\mathsf{Eval}_{\mathsf{sk}}(m) = \mathsf{Eval}_{\mathsf{sk}}(m')$, breaking unpredictability. In our setting, we opt for trusted setup, as it is unclear how to construct a prime p and curve E_0 such that the conditions of (2) are satisfied. We stress that if an efficient testing procedure for (2) is found, a public parameter could consist of a curve, and a representation of its endomorphism ring, so that users may check the parameters meet the conditions and therefore $\mathsf{CGL}(E_0, \cdot)$ is injective, and the underlying VRF will satisfy full unique provability. We leave this as future work.

Conjectured hardness of the one-more evaluation problem. Given that the function $\mathsf{CGL}(E_0, \cdot)$ is pre-image and collision resistant, there are no trivial ways to break the one-more evaluation problem via either recovering the secret key from the public key, or finding collisions in the message evaluation. We further justify why the functions outputs are unpredictable, that is, why for possibly related messages m, m', the outputs $\mathsf{CGL}(\mathsf{CGL}(E_0, m), k)$ and $\mathsf{CGL}(\mathsf{CGL}(E_0, m'), k)$ appear uncorrelated.

Although the VRF evaluator reuses the same key directing walks on the supersingular isogeny graph, starting at different curves $E_m, E_{m'}$, there is no real algebraic connection between different evaluations of the VRF. At the $(i+1)$th-step of the secret walk, given that the i-th curve is $E_i : y^2 = x^3 + A_i x^2 + C_i x$, the direction of the walk dictated by the bits of k is determined by which root of $\mathrm{Re}(\sqrt{C_i})$ is a quadratic residue modulo p. This is a completely arbitrary choice of ordering, and depends on the underlying curve at every step. This makes it difficult to correlate different evaluations of the VRF under the same key.

Furthermore, one can quickly convince oneself that $\mathsf{CGL}(\mathsf{CGL}(E_0, m), k) = \mathsf{CGL}(E_0, m \parallel k)$, if m and k are written with the least significant bit first. Consider the tree of walks starting at the curve E_0 dictated by $m \parallel k$ for all $m \in \{-1, 1\}^n$, $k \in \{-1, 1\}^e$. Such a tree, rooted at E_0, may be viewed as a depth $n + e$ subtree (with a missing branch) of covering graph of the 2-supersingular isogeny graph: the 2-adic Bruhat-Tits tree (see [3] for details of this correspondence). Observe that if a single bit of a message is different, then the walk dictated by the remaining string is in a completely different branch of the tree. The covering map from ℓ-adic Bruhat-Tits trees to the ℓ-supersingular isogeny graph, which allows one to efficiently translate between vertices in the tree and graph, is believed to be hard to compute when the endomorphism ring of E_0 is unknown.

Secure parameters. In light of the previous discussion, we propose parameters for the VRF in this section. Since generic proofs scale unfavourably with the size of the underlying field of operation; we opt for $\log p \approx 2\lambda$, and an E_0 of unknown endomorphism ring; in the security model where collisions can exist, but must be computationally hard to find.

We opt to set the lengths of the message and key walks to be equal and long enough to ensure hardness of the isogeny problem ($n \approx 2\lambda$).

Given these considerations, we state our security assumption below.

Conjecture 1 (Radical CGL one-more evaluation problem). The one-more evaluation problem from Definition 4 is hard, when $(\mathcal{W}, \mathcal{E}, \mathcal{K}, E_0) \leftarrow \mathsf{SetUp}(1^\lambda)$ is instantiated as follows.

- $\mathcal{W}$ is the radical CGL hash function CGL from Figure 1,
- $\mathcal{E}$ is the set of supersingular elliptic curves defined over the finite field $\mathbb{F}_{p^2}$, where $\log p \approx 2\lambda$,
- $\mathcal{K}$ is the set of binary strings $\{-1, 1\}^n$ of fixed length $n \geq 2\lambda$, and
- $E_0 \in \mathcal{E}$ is a starting elliptic curve of unknown endomorphism ring.

4.2.2 Parameter and proof system selection. Due to the restrictions of our proof system from Section 3, we want to work with fields of characteristic $p \equiv 3 \pmod 4$ such as, for example, quasi-Mersenne primes of the form $p = c2^e - 1$. See the table of proposed parameters for NIST level I security in Table 3. We choose our prime in order for their representations to require the minimal number of 32-bit words (i.e. $\log p$ fits in the minimal multiple of 32).

p	n	$\lceil \log p \rceil$
$5 \cdot 2^{248} - 1$	256	251

Table 3: Proposed parameters for the CGL VRF for NIST level I security ($\lambda = 128$).

We require post-quantum generic proof systems for R1CS that support $\mathbb{F}_p$-arithmetic with $p \equiv 3 \pmod 4$. This implies that $\mathbb{F}_p^\times$ has two-adicity of one, so we cannot use Reed-Solomon based SNARKs (such as those based on FRI [10]), which require two-adicity logarithmic in the size of the circuit. Fortunately, recent developments in SNARK literature have introduced *field agnostic* SNARKs for the context of proving evaluations of the ECDSA verification circuit, which operate over $p = 3 \pmod 4$.

Brakedown [49] (its improvements and variants [14,50]), Orion [67] and BaseFold [68] fit our constraints and are viable candidates to instantiate our NIZK. We choose the protocol of [14] as it is the current state-of-the-art by all performance metrics.

Unfortunately, none of these sources have made a working implementation available which both allows for arbitrary primes and zero-knowledge; so we must rely on the benchmarked costs of the 256-bit prime used in [14]. Note that their implementation (and indeed, all of the listed candidate SNARKs) do not provide zero-knowledge. It appears there are two solutions to achieving zero-knowledge: (1) apply a recursive proof composition technique with a zero-knowledge argument, which are not field agnostic, and thus not suitable for our application; or (2) apply the standard techniques of [18], which appear to be appropriate for this setting. We cannot provide a concrete estimate, but these measures are unlikely to cause substantial overhead.

For path lengths of $n = 256$, we end up with 4352 constraints, and hence base our results on circuits of size 2^{13}. We find the costs outlined in Table 1.

5 A verifiable random function from CSIDH

In this section, we introduce a new construction based on the group action description from CSIDH [22]. In particular, we work in the effective group action (EGA) setting introduced in Section 2.2.

We note that two VRFs in the EGA setting have recently been proposed by Lai [54]. Lai's constructions rely on translations of the Naor-Reingold PRF [59] to the group action setting. The downside of this type of construction is that the public key size, proof (and verification) runtime and proof size all scale quadratically in the security parameter λ.

Instead, our approach is inspired by the OPRF construction from [36], which considerably improves upon the Naor-Reingold based OPRF from [51] by using a completely different approach of *polynomial evaluation in the exponent.* The latter means to evaluate the function

$$[f(m)]E_0 \tag{9}$$

jointly between the client with input $m \in \mathbb{Z}_N$ and the server with input $f(X) \in \mathbb{Z}_N[X]$, where $N = \#\mathsf{cl}(\mathcal{O})$. The result is a two-round protocol, rather than λ rounds, with a reduction by a factor λ in both the computational cost and public key size, when compared to [51]. Due to the regularity of the group action, any collision-resistant polynomial f with output distribution indistinguishable from uniform is enough to guarantee security of the OPRF. The caveat, however, is that the protocol either relies on a trusted or a computationally heavy setup phase that has to be run once for each client-server pair.

We use an approach inspired by [36], where our VRF output is exactly the same as in equation (9) for a given input $m \in \mathbb{Z}_N$. However, since we do not need to mask the message as in an OPRF, our proof of correct evaluation by the server takes a very different approach, in particular, one which doesn't need any initial (trusted) setup phase. Our construction relies on the following hardness assumption, introduced in [36].

Problem 5 (One-more unpredictability). It is hard for an adversary with oracle access to $[f(m)]E_0 \leftarrow \mathsf{VRF}_f(m)$ to find a pair $(m^*, [f(m^*)]E_0)$, where m^* has not been queried to VRF_f before.

The security of this assumption is discussed in [36]. In fact, due to the group action being regular, the only properties we need from f are uniformly random output (or at least indistinghuishability from uniform) and collision-resistance. As discussed in [36], permutation polynomials are perfect candidates. Furthermore, choosing an f of low degree is also beneficial from a complexity point of view. We note, however, that if f has degree $d = 1$, our construction is secure only if the queries to the VRF_f-oracle are classical. In case where quantum superposition queries to VRF_f are allowed, the degree needs to be $d \geq 2$. Linear polynomials are insecure in this case, since the attacker can use the quantum reduction of computational Diffie-Hellman to key recovery introduced in [44] to recover the secret polynomial. Going to $d = 2$, we note that permutation polynomials only exist over characteristic 2, but 2-to-1 polynomials are still heuristically secure [36]. For $d = 3$, all of these issues are solved, but we go for slightly higher computational and communication complexity.

Remark 3. We point out that due to N being composite, a malicious client could potentially query an input m dividing N and effectively lets $f(X)$ act in a subgroup of $\mathbb{Z}_N$, which might reveal information about $f(X)$ [36]. This effectively reduces security to the largest prime divisor of N, as pointed out in [55]. The current CSI-FiSh parameter set [13] would imply classical security of 70 bits only. Thankfully, the recent results from [61] present a constructive method that allows to create class groups of any order $c - 1$, where c is a prime (c is a conductor of the Gaussian integers, see [61] for more details). By choosing $c = 2^{257} + 269451$, we find that $c - 1$ is divided only by 2 and a 256-bit prime, which implies a 128 bit security level.

5.1 Our construction

The idea behind our VRF is that the server samples and commits to a polynomial $f(X) \in \mathbb{Z}_N[X]$ of degree $d = \deg f$ as their public key, and then later proves that it used the same polynomial $f(X)$ in the evaluation of the client's input m. To this end, we define the public key to have the form

$$\mathsf{pk} = ([f(0)]E_0, [f(1)]E_0, f(2), \ldots, f(d)) .$$

We note that we can safely disclose $f(2), \ldots, f(d)$, since knowledge of less than $d + 1$ shares information-theoretically hides the secret polynomial $f(X)$. In theory, we could further disclose $f(1)$, yet we will show in Section 5.1.1 below that doing so would violate the one-more unpredictability of our construction. The reason why we reveal $f(2), \ldots, f(d)$ in the first place and not e.g. have $[f(2)]E_0, \ldots, [f(d)]E_0$ in our public key is simply that this results in a faster proof system.

First of all, we note that the $d + 1$ evaluations in the public key uniquely define a polynomial of degree d. In the case of $[f(0)]E_0$ and $[f(1)]E_0$, this is guaranteed by the regularity of the group action. Now, proving correct evaluation $E_m = [f(m)]E_0$ of an input m can be done using a proof for the following relation

$$\exists f(X) \in \mathbb{Z}_N[X] \text{ with } \deg f \leq d :$$

$$\mathsf{pk} = ([f(0)]E_0, [f(1)]E_0, f(2), \ldots, f(d)) \wedge E_m = [f(m)]E_0 \,. \tag{10}$$

We note that this relation is quite similar to the statement of the piecewise verifiable proofs (PVPs) introduced in [12], which for a statement $(x_0, \ldots, x_n)$, where $n \geq d$, tries to prove relations of the type

$$\exists f(X) \in \mathbb{Z}_N[X] \text{ with } \deg f \leq d : \\ x_0 = [f(0)]E_0 \wedge x_1 = \mathcal{C}(f(1), y_1) \wedge \cdots \wedge x_n = \mathcal{C}(f(n), y_n) \,,$$

where $\mathcal{C}$ are commitment functions and $y_1, \ldots, y_n$ are randomizers. Thus our proof system that we present in Figure 3 below is somewhat inspired by the PVP constructions, but differs in a few points. The most important differences are that we have three polynomial evaluations applied to elliptic curves, $[f(0)]E_0, [f(1)]E_0$ and $[f(m)]E_0$ rather than just one, and that we do not need commitment schemes to hide the evaluations of $f(2), \ldots, f(d)$. In Figure 3, $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\lambda$ denotes a random oracle. Using this proof system, we can then make the server's evaluation verifiable. We present our VRF in Figure 4, for which we further define the random oracle $H : \{0,1\}^* \to \{0,1\}^{2\lambda}$.

```
CNIZK.P_sk(pk, m, E_m):
    1. Parse $\mathsf{pk} = (\mathsf{P}_0 = [f(0)]E_0, \mathsf{P}_1 = [f(1)]E_1, f(2), \ldots, f(d))$
    2. For $j = 1, \ldots, \lambda$:
       (a) Sample $b_j(X) \leftarrow \mathbb{Z}_N[X]$ of degree $d$
       (b) Compute $S_j = ([b_j(0)]E_0, [b_j(1)]E_0, b_j(2), \ldots, b_j(d), [b_j(m)]E_0)$
    3. $c_1 \ldots c_\lambda \leftarrow \mathsf{H}(S_1, \ldots, S_\lambda)$
    4. For $j = 1, \ldots, \lambda$, compute $r_j(X) = b_j(X) - c_j f(X)$
    5. Return $(c_1, \ldots, c_\lambda), (r_1(X), \ldots, r_\lambda(X))$

CNIZK.V(pk, m, E_m, π):
    1. Parse $\mathsf{pk} = (\mathsf{P}_0, \mathsf{P}_1, f(2), \ldots, f(d))$
       and $\pi = ((c_1, \ldots, c_\lambda), (r_1(X), \ldots, r_\lambda(X)))$
    2. For $j = 1, \ldots, \lambda$:
       if $c_j = 0$: compute $\hat{S}_j = ([r_j(0)]E_0, [r_j(1)]E_0, r_j(2), \ldots, r_j(d), [r_j(m)]E_0)$
       if $c_j = 1$: compute
       $\hat{S}_j = ([r_j(0)]\mathsf{P}_0, [r_j(1)]\mathsf{P}_1, r_j(2) + f(2), \ldots, r_j(d) + f(d), [r_j(m)]E_m)$
    3. Return $c_1 \ldots c_\lambda \overset{?}{=} \mathsf{H}(\hat{S}_1, \ldots, \hat{S}_\lambda)$
```

Fig. 3: Proof system for our CSIDH-based VRF.

We prove the following theorems in Appendix B.2

Theorem 2. *The* CNIZK *proof system from Figure 3 is a non-interactive zero-knowledge proof of knowledge in the QROM.*

Theorem 3. *Let* CNIZK *be a NIZKPoK for relation 10, then Figure 4 is a verifiable pseudorandom function (VRF) in the random oracle model.*

```
$\mathbf{Setup}(1^\lambda)$:
    1. Return an EGA $(\mathbb{Z}_N, \mathcal{E}, E_0)$ with $\mathcal{E}$ the set of supersingular elliptic curves
       over $\mathbb{F}_p$ with origin $E_0$, and where $p$ is defined by the security parameter $\lambda$.
$\mathbf{KeyGen}_d(\mathsf{pp})$:
    1. Sample $\mathsf{sk} = f(X) \leftarrow \mathbb{Z}_N[X]$ of degree $d$ with non-zero coefficients.
    2. Compute $\mathsf{pk} = ([f(0)]E_0, [f(1)]E_0, f(2), \ldots, f(d))$.
    3. Return $(\mathsf{sk}, \mathsf{pk})$
$\mathbf{Eval}_f(m, \mathsf{pk}; \mathsf{pp})$:
    1. Compute $E_m = [f(m)]E_0$
    2. Run $\pi \leftarrow \mathsf{CNIZK}.P_{\mathsf{sk}}(\mathsf{pk}, m, E_m)$.
    3. Return $h = H(\mathsf{pk}, m, E_m)$ and the proof $(E_m, \pi)$.
$\mathbf{Verify}_{\mathsf{pk}}(m, h, (E_m, \pi); \mathsf{pp})$:
    1. Return $h \stackrel{?}{=} H(\mathsf{pk}, m, E_m) \ \wedge\ \mathsf{CNIZK}.V(\mathsf{pk}, m, E_m, \pi)$
```

Fig. 4: Verifiable pseudorandom function from polynomial evaluation in the exponent.

5.1.1 Interpolation attack. In this section, we argue why we can't reveal $f(1)$ as part of the public key, but only $[f(1)]E_0$, even though this increases the cost of the CNIZK protocol. The short answer is that disclosing $f(1)$ actually reveals enough information to an attacker $\mathcal{A}$ to break residual pseudorandomness.

To see this, assume we already know the evaluation of $\mathbf{Eval}_f(m)$, which includes the curve $[f(m)]E_0$ in its proof. In theory, using Lagrange interpolation, any evaluation $[f(m')]E_0$ could now be computed as

$$\Big[\sum_{i\in S} \ell_i^S(m')f(i)\Big]E_0 = \Big[\sum_{i\in S\setminus\{m\}} \ell_i^S(m')f(i)\Big]\Big[\ell_m^S(m')f(m)\Big]E_0\,,$$

where $S = \{1, \ldots, d, m\}$ and where the Lagrange basis polynomials are defined as

$$\ell_i^S(X) = \prod_{j\in S\setminus\{i\}} \frac{X-j}{i-j}\,.$$

While we can't simply turn $[f(m)]E_0$ into $[\ell_m^S(m')f(m)]E_0$, we can however look at the cases, where $\ell_m^S(m') = 1$, since we could then readily apply the interpolation formula on $[f(m)]E_0$ and get a new evaluation $[f(m')]E_0$ in this way. Since the Lagrange basis polynomials $\ell_i^S(X)$ have degree $|S| - 1 = d$, we will in general find d solutions for possible evaluations $[f(m')]E_0$, only one of which is $m' = m$ and up to $d-1$ allowing us to generate a new evaluation $H(m', [f(m')]E_0)$ that has not been queried to $\mathbf{Eval}_f$ before. The attack proceeds as follows.

1. First, $\mathcal{A}$ queries $\mathbf{Eval}_f(m)$ for some message m to learn $E_m = [f(m)]E_0$.

2. Now, let $\{\ell_i^S(X)\}_{i\in S}$ be the Lagrange basis polynomials with respect to the set $S = \{1, \dots, d, m\}$.
3. $\mathcal{A}$ chooses one of the solutions $X = m'$ to $\ell_m^S(X) = 1$ distinct from m.
4. Finally, $\mathcal{A}$ computes

$$[f(m')]E_0 = \Big[\sum_{i\in S\setminus\{m\}} \ell_i^S(m')f(i)\Big][f(m)]E_0$$

and returns $H(m', [f(m')]E_0)$.

This attack can be thwarted simply by disclosing at most $d-1$ shares of $f(X)$ in the public key, which is why we chose the form defined above. In this case, the largest interpolation set S we could use would have size $|S| = d$, which is not enough information to correctly interpolate.

Example 1 (Attacking $d = 2$). Assume the public key is $\mathsf{pk} = ([f(0)]E_0, f(1), f(2))$ and we have queried m and received the element $[f(m)]E_0$. Solving

$$\ell_m^{\{1,2,m\}}(m') = \frac{m'-1}{m-1}\frac{m'-2}{m-2} = 1$$

gives us $(m'-1)(m'-2) = (m'-1)(m'-2)$, which has solutions $m' = m$ and $m' = -m+3$. By choosing the latter, the attacker could compute

$$\left[\ell_1^{\{1,2,m\}}(m')f(1) + \ell_2^{\{1,2,m\}}(m')f(2)\right][f(m)]E_0 = [f(m')]E_0\,,$$

which allows to compute a correct evaluation $(m', H(\mathsf{pk}, m', [f(m')]E_0))$ for $m' = -3m+1$ for each m previously queried.

Interestingly, for $d = 1$, this is not a problem, as the Lagrange basis polynomials are linear and thus $\ell_i^S(X) = 1$ would only have a single solution, which is $m' = m$. We discuss how this impacts our protocol for $d = 1$ below. We note that we do have to take care that all polynomial coefficients are non-zero in this case, since for $f(X) = f_0 + f_1X$, if $f_0 = 0$, then twisting allows to compute an evaluation for $-m$ after querying m. If $f_1 = 0$, then the VRF is trivial.

5.1.2 The simplified case for $d = 1$.

The discussion from the previous section allows us to simplify public keys in the linear case to

$$\mathsf{pk} = ([f(0)]E_0, f(1))\,.$$

With this modification, we also get a simpler relation

$$\exists f(X) \in \mathbb{Z}_N[x] \text{ with } \deg f = 1 : \mathsf{pk} = ([f(0)]E_0, f(1)) \wedge E_m = [f(m)]E_0$$

and a simpler proof system. The modification of Figure 3 to this end is straightforward. We note that this impacts the runtimes of the CNIZK proof in a beneficial way, i.e. the computations of S_j and $\hat{S}_j$ are now performed with only two group actions instead of three. We discuss costs later in Section 5.2, as we have another optimization up our sleeve.

5.1.3 Optimization using twists. When we work with symmetric EGAs from Definition 2, we can increase the challenge space of CNIZK from 2 to 3 per round using the *twist trick* first proposed in [33]. In order to see this, note that the following values are recoverable from the public key

$$[-f(0)]E_0, [-f(1)]E_0, -f(2), \dots, -f(d)\,,$$

by using the twisting operation on the elliptic curves and negating the polynomial evaluations.[6] We can do the same to also compute $[-f(m)]E_0$. Thus, instead of sampling the challenges from $\{0,1\}$, we can now sample them from $\{-1,0,1\}$, forcing the prover to reveal $r(X) = b(X) + f(X)$ when $c = -1$ and letting the verifier check whether this connects the commitments to the "negative part" of the public key. This effectively decreases the soundness error per round from $1/2$ to $1/3$ and consequently the number of rounds from λ repetitions to $\kappa = \lceil \lambda / \log_2 3 \rceil$.

The protocol in Figure 3 is easily modified to account for this. Simply replace λ by κ and the hash function by $\mathsf{H} : \{0,1\}^* \to \{-1,0,1\}^\kappa$. Note that the responses stay the same.

5.2 Cost overview

We express the computational cost of our VRF designs by counting the number of group actions, as other costs are negligible in comparison. The dominating factor in both computational and communication costs are the proofs. We find that CNIZK has output size $\kappa + (d+1)\kappa \log N$ bits, where $\kappa \in \{\lambda, \lceil \lambda/\log_2 3 \rceil\}$, depending on whether we can apply the twist trick or not. The number of group actions to compute is 3κ group actions for $d \geq 2$ to both prove and verify, and 2κ for $d = 1$.

Using these numbers, we express the full output size (in bits) and computational cost (in number of group actions) of **Eval** and **Verif** in the table below. We also add the computational cost of computing **KeyGen** as well as the resulting public key size. The secret key has $(d+1)\log N$ bits, which can be reduced to a seed of λ bits, as is done e.g. in [13]. We use the placeholder constant c to express the size of elliptic curve representations, where $c = 1$ in the classical CSI-FiSh [13] type instantiation and $c = 6$ when using oriented elliptic curves, such as in KLaPoTi [61].

	KeyGen	**Eval**	**Verif**
Output size ($d = 1$)	$c \log p + \log N$	$\kappa + (d+1)\kappa \log N + c \log p + 2\lambda$	—
Output size ($d \geq 2$)	$2c \log p + (d-1) \log N$		—
Group actions ($d = 1$)	1	$2\kappa + 1$	2κ
Group actions ($d \geq 2$)	2	$3\kappa + 1$	3κ

[6] Note that in the case of linear polynomials in symmetric EGAs, $[f(2)]E_0$ is leaked via $[2f(1)][-f(0)]E_0$. This may be treated as a public parameter in the same way that $[f(2)]E_0, \dots, [f(d)]E_0$ are leaked through public keys for larger d.

We instantiate our results using the EGA from KLaPoTi [61] in Table 2 and compare our results to the previous state of the art Capybara and Tsubaki from [54], based on the EGA from [13].

Acknowledgements. This work was supported in part by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement ISOCRYPT - No. 101020788), by CyberSecurity Research Flanders with reference number VR20192203 and by the research grant VIL53029 from VILLUM FONDEN. This work was also supported by a grant from the Ministry of Business, Innovation and Employment, New Zealand.

The authors would like to thank Wouter Castryck, Steven Galbraith and Frederik Vercauteren for helpful discussions regarding this work.

References

1. Alamati, N., De Feo, L., Montgomery, H., Patranabis, S.: Cryptographic group actions and applications. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 411–439. Springer, Cham (Dec 2020). https://doi.org/10.1007/978-3-030-64834-3_14
2. Ames, S., Hazay, C., Ishai, Y., Venkitasubramaniam, M.: Ligero: Lightweight sublinear arguments without a trusted setup. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017. pp. 2087–2104. ACM Press (Oct / Nov 2017). https://doi.org/10.1145/3133956.3134104
3. Amorós, L., Iezzi, A., Lauter, K., Martindale, C., Sotáková, J.: Explicit Connections Between Supersingular Isogeny Graphs and Bruhat–Tits Trees, pp. 39–73. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-77700-5_2, https://doi.org/10.1007/978-3-030-77700-5_2
4. Atapoor, S., Baghery, K., Cozzo, D., Pedersen, R.: Practical robust DKG protocols for CSIDH. In: Tibouchi, M., Wang, X. (eds.) ACNS 23International Conference on Applied Cryptography and Network Security, Part II. LNCS, vol. 13906, pp. 219–247. Springer, Cham (Jun 2023). https://doi.org/10.1007/978-3-031-33491-7_9
5. Atapoor, S., Baghery, K., Cozzo, D., Pedersen, R.: VSS from distributed ZK proofs and applications. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part I. LNCS, vol. 14438, pp. 405–440. Springer, Singapore (Dec 2023). https://doi.org/10.1007/978-981-99-8721-4_13
6. Baghery, K., Cozzo, D., Pedersen, R.: An isogeny-based ID protocol using structured public keys. In: Paterson, M.B. (ed.) 18th IMA International Conference on Cryptography and Coding. LNCS, vol. 13129, pp. 179–197. Springer, Cham (Dec 2021). https://doi.org/10.1007/978-3-030-92641-0_9
7. Basso, A.: A post-quantum round-optimal oblivious PRF from isogenies. In: Carlet, C., Mandal, K., Rijmen, V. (eds.) SAC 2023. LNCS, vol. 14201, pp. 147–168. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-53368-6_8

8. Basso, A., Codogni, G., Connolly, D., De Feo, L., Fouotsa, T.B., Lido, G.M., Morrison, T., Panny, L., Patranabis, S., Wesolowski, B.: Supersingular curves you can trust. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part II. LNCS, vol. 14005, pp. 405–437. Springer, Cham (Apr 2023). https://doi.org/10.1007/978-3-031-30617-4_14
9. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V. (eds.) ACM CCS 93. pp. 62–73. ACM Press (Nov 1993). https://doi.org/10.1145/168588.168596
10. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Fast reed-solomon interactive oracle proofs of proximity. In: Chatzigiannakis, I., Kaklamanis, C., Marx, D., Sannella, D. (eds.) ICALP 2018. LIPIcs, vol. 107, pp. 14:1–14:17. Schloss Dagstuhl (Jul 2018). https://doi.org/10.4230/LIPIcs.ICALP.2018.14
11. Ben-Sasson, E., Chiesa, A., Riabzev, M., Spooner, N., Virza, M., Ward, N.P.: Aurora: Transparent succinct arguments for R1CS. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 103–128. Springer, Cham (May 2019). https://doi.org/10.1007/978-3-030-17653-2_4
12. Beullens, W., Disson, L., Pedersen, R., Vercauteren, F.: CSI-RAShi: Distributed key generation for CSIDH. In: Cheon, J.H., Tillich, J.P. (eds.) Post-Quantum Cryptography - 12th International Workshop, PQCrypto 2021. pp. 257–276. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-81293-5_14
13. Beullens, W., Kleinjung, T., Vercauteren, F.: CSI-FiSh: Efficient isogeny based signatures through class group computations. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part I. LNCS, vol. 11921, pp. 227–247. Springer, Cham (Dec 2019). https://doi.org/10.1007/978-3-030-34578-5_9
14. Block, A.R., Fang, Z., Katz, J., Thaler, J., Waldner, H., Zhang, Y.: Field-Agnostic SNARKs from Expand-Accumulate Codes. In: Reyzin, L., Stebila, D. (eds.) Advances in Cryptology - CRYPTO 2024. pp. 276–307. Springer Nature Switzerland, Cham (2024)
15. Bodaghi, O., Safavi-Naini, R.: Short Paper: Breaking X-VRF, a Post-Quantum Verifiable Random Function. In: Financial Crypto 2024 (March 2024), available at https://fc24.ifca.ai/preproceedings/213.pdf
16. Boneh, D., Kogan, D., Woo, K.: Oblivious pseudorandom functions from isogenies. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 520–550. Springer, Cham (Dec 2020). https://doi.org/10.1007/978-3-030-64834-3_18
17. Bonnetain, X., Schrottenloher, A.: Quantum security analysis of CSIDH. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part II. LNCS, vol. 12106, pp. 493–522. Springer, Cham (May 2020). https://doi.org/10.1007/978-3-030-45724-2_17
18. Bootle, J., Cerulli, A., Ghadafi, E., Groth, J., Hajiabadi, M., Jakobsen, S.K.: Linear-time zero-knowledge proofs for arithmetic circuit satisfiability. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part III. LNCS, vol. 10626, pp. 336–365. Springer, Cham (Dec 2017). https://doi.org/10.1007/978-3-319-70700-6_12
19. Buser, M., Dowsley, R., Esgin, M.F., Kasra Kermanshahi, S., Kuchta, V., Liu, J.K., Phan, R.C.W., Zhang, Z.: Post-quantum verifiable random function from symmetric primitives in PoS blockchain. In: Atluri, V., Di Pietro, R., Jensen, C.D., Meng, W. (eds.) ESORICS 2022, Part I. LNCS, vol. 13554, pp. 25–45. Springer, Cham (Sep 2022). https://doi.org/10.1007/978-3-031-17140-6_2

20. Castryck, W., Decru, T.: CSIDH on the surface. In: Ding, J., Tillich, J.P. (eds.) Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020. pp. 111–129. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-44223-1_7
21. Castryck, W., Decru, T., Vercauteren, F.: Radical isogenies. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 493–519. Springer, Cham (Dec 2020). https://doi.org/10.1007/978-3-030-64834-3_17
22. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: An efficient post-quantum commutative group action. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part III. LNCS, vol. 11274, pp. 395–427. Springer, Cham (Dec 2018). https://doi.org/10.1007/978-3-030-03332-3_15
23. Charles, D.X., Lauter, K.E., Goren, E.Z.: Cryptographic hash functions from expander graphs. Journal of Cryptology **22**(1), 93–113 (Jan 2009). https://doi.org/10.1007/s00145-007-9002-x
24. Chen, J., Micali, S.: Algorand: A secure and efficient distributed ledger. Theor. Comput. Sci. **777**, 155–183 (2019). https://doi.org/10.1016/J.TCS.2019.02.001
25. Chen, M., Leroux, A., Panny, L.: SCALLOP-HD: Group action from 2-dimensional isogenies. In: Tang, Q., Teague, V. (eds.) PKC 2024, Part II. LNCS, vol. 14603, pp. 190–216. Springer, Cham (Apr 2024). https://doi.org/10.1007/978-3-031-57725-3_7
26. Choi, K., Manoj, A., Bonneau, J.: SoK: Distributed randomness beacons. In: 2023 IEEE Symposium on Security and Privacy. pp. 75–92. IEEE Computer Society Press (May 2023). https://doi.org/10.1109/SP46215.2023.10179419
27. Cong, K., Lai, Y.F., Levin, S.: Efficient isogeny proofs using generic techniques. In: Tibouchi, M., Wang, X. (eds.) ACNS 23International Conference on Applied Cryptography and Network Security, Part II. LNCS, vol. 13906, pp. 248–275. Springer, Cham (Jun 2023). https://doi.org/10.1007/978-3-031-33491-7_10
28. Couveignes, J.M.: Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291 (2006), https://eprint.iacr.org/2006/291
29. Dartois, P., Leroux, A., Robert, D., Wesolowski, B.: SQIsignHD: New dimensions in cryptography. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part I. LNCS, vol. 14651, pp. 3–32. Springer, Cham (May 2024). https://doi.org/10.1007/978-3-031-58716-0_1
30. De Feo, L.: Mathematics of isogeny based cryptography. arXiv (2017), http://arxiv.org/abs/1711.04062
31. De Feo, L., Dobson, S., Galbraith, S.D., Zobernig, L.: SIDH proof of knowledge. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part II. LNCS, vol. 13792, pp. 310–339. Springer, Cham (Dec 2022). https://doi.org/10.1007/978-3-031-22966-4_11
32. De Feo, L., Fouotsa, T.B., Kutas, P., Leroux, A., Merz, S.P., Panny, L., Wesolowski, B.: SCALLOP: Scaling the CSI-FiSh. In: Boldyreva, A., Kolesnikov, V. (eds.) PKC 2023, Part I. LNCS, vol. 13940, pp. 345–375. Springer, Cham (May 2023). https://doi.org/10.1007/978-3-031-31368-4_13
33. De Feo, L., Galbraith, S.D.: SeaSign: Compact isogeny signatures from class group actions. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part III. LNCS, vol. 11478, pp. 759–789. Springer, Cham (May 2019). https://doi.org/10.1007/978-3-030-17659-4_26
34. De Feo, L., Jao, D., Plût, J.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. J. Math. Cryptol. **8**(3), 209–247 (2014). https://doi.org/10.1515/jmc-2012-0015

35. De Feo, L., Kohel, D., Leroux, A., Petit, C., Wesolowski, B.: SQISign: Compact post-quantum signatures from quaternions and isogenies. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part I. LNCS, vol. 12491, pp. 64–93. Springer, Cham (Dec 2020). https://doi.org/10.1007/978-3-030-64837-4_3
36. Delpech de Saint Guilhem, C., Pedersen, R.: New proof systems and an OPRF from CSIDH. In: Tang, Q., Teague, V. (eds.) PKC 2024, Part II. LNCS, vol. 14603, pp. 217–251. Springer, Cham (Apr 2024). https://doi.org/10.1007/978-3-031-57725-3_8
37. Diamond, B.E., Posen, J.: Proximity testing with logarithmic randomness. CiC **1**(1), 2 (2024). https://doi.org/10.62056/aksdkp10
38. Doliskani, J., Pereira, G.C.C.F., Barreto, P.S.L.M.: Faster cryptographic hash function from supersingular isogeny graphs. In: Smith, B., Wu, H. (eds.) SAC 2022. LNCS, vol. 13742, pp. 399–415. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-58411-4_18
39. Don, J., Fehr, S., Majenz, C., Schaffner, C.: Security of the Fiat-Shamir transformation in the quantum random-oracle model. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part II. LNCS, vol. 11693, pp. 356–383. Springer, Cham (Aug 2019). https://doi.org/10.1007/978-3-030-26951-7_13
40. Esgin, M.F., Ersoy, O., Kuchta, V., Loss, J., Sakzad, A., Steinfeld, R., Yang, X., Zhao, R.K.: A new look at blockchain leader election: Simple, efficient, sustainable and post-quantum. In: Liu, J.K., Xiang, Y., Nepal, S., Tsudik, G. (eds.) Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security, ASIA CCS 2023, Melbourne, VIC, Australia, July 10-14, 2023. pp. 623–637. ACM (2023). https://doi.org/10.1145/3579856.3595792
41. Esgin, M.F., Kuchta, V., Sakzad, A., Steinfeld, R., Zhang, Z., Sun, S., Chu, S.: Practical post-quantum few-time verifiable random function with applications to algorand. In: Borisov, N., Díaz, C. (eds.) FC 2021, Part II. LNCS, vol. 12675, pp. 560–578. Springer, Berlin, Heidelberg (Mar 2021). https://doi.org/10.1007/978-3-662-64331-0_29
42. Esgin, M.F., Steinfeld, R., Liu, D., Ruj, S.: Efficient hybrid exact/relaxed lattice proofs and applications to rounding and VRFs. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part V. LNCS, vol. 14085, pp. 484–517. Springer, Cham (Aug 2023). https://doi.org/10.1007/978-3-031-38554-4_16
43. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 186–194. Springer, Berlin, Heidelberg (Aug 1987). https://doi.org/10.1007/3-540-47721-7_12
44. Galbraith, S., Panny, L., Smith, B., Vercauteren, F.: Quantum Equivalence of the DLP and CDHP for Group Actions. Mathematical Cryptology **1**(1), 40–44 (2021)
45. Galbraith, S.D.: Constructing isogenies between elliptic curves over finite fields. LMS Journal of Computation and Mathematics **2**, 118–138 (1999)
46. Giunta, E., Stewart, A.: Unbiasable verifiable random functions. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part IV. LNCS, vol. 14654, pp. 142–167. Springer, Cham (May 2024). https://doi.org/10.1007/978-3-031-58737-5_6
47. Goldberg, S., Naor, M., Papadopoulos, D., Reyzin, L., Vasant, S., Ziv, A.: NSEC5: Provably preventing DNSSEC zone enumeration. In: NDSS 2015. The Internet Society (Feb 2015). https://doi.org/10.14722/ndss.2015.23211
48. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM Journal on Computing **18**(1), 186–208 (1989)

49. Golovnev, A., Lee, J., Setty, S.T.V., Thaler, J., Wahby, R.S.: Brakedown: Linear-time and field-agnostic SNARKs for R1CS. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part II. LNCS, vol. 14082, pp. 193–226. Springer, Cham (Aug 2023). https://doi.org/10.1007/978-3-031-38545-2_7
50. Haböck, U.: Brakedown's expander code. Cryptology ePrint Archive, Report 2023/769 (2023), https://eprint.iacr.org/2023/769
51. Heimberger, L., Hennerbichler, T., Meisingseth, F., Ramacher, S., Rechberger, C.: Oprfs from isogenies: Designs and analysis. In: Zhou, J., Quek, T.Q.S., Gao, D., Cárdenas, A.A. (eds.) Proceedings of the 19th ACM Asia Conference on Computer and Communications Security, ASIA CCS 2024, Singapore, July 1-5, 2024. ACM (2024). https://doi.org/10.1145/3634737.3645010
52. den Hollander, T., Kleine, S., Mula, M., Slamanig, D., Spindler, S.A.: More efficient isogeny proofs of knowledge via canonical modular polynomials. Cryptology ePrint Archive, Paper 2024/1738 (2024), https://eprint.iacr.org/2024/1738
53. Kohel, D., Lauter, K.E., Petit, C., Tignol, J.: On the quaternion ℓ-isogeny path problem. LMS Journal of Computation and Mathematics **17**, 418–432 (2014)
54. Lai, Y.F.: CAPYBARA and TSUBAKI: Verifiable random functions from group actions and isogenies. Cryptology ePrint Archive, Report 2023/182 (2023), https://eprint.iacr.org/2023/182
55. Lai, Y.F.: A note on isogeny group action-based pseudorandom functions. Cryptology ePrint Archive, Paper 2024/2042 (2024), https://eprint.iacr.org/2024/2042
56. Leroux, A.: Verifiable random function from the deuring correspondence and higher dimensional isogenies. Cryptology ePrint Archive, Report 2023/1251 (2023), https://eprint.iacr.org/2023/1251
57. Micali, S., Rabin, M.O., Vadhan, S.P.: Verifiable random functions. In: 40th FOCS. pp. 120–130. IEEE Computer Society Press (Oct 1999). https://doi.org/10.1109/SFFCS.1999.814584
58. Mokrani, Y., Jao, D.: Generating supersingular elliptic curves over $\mathbb{F}_p$ with unknown endomorphism ring. In: Chattopadhyay, A., Bhasin, S., Picek, S., Rebeiro, C. (eds.) INDOCRYPT 2023, Part I. LNCS, vol. 14459, pp. 159–174. Springer, Cham (Dec 2023). https://doi.org/10.1007/978-3-031-56232-7_8
59. Naor, M., Reingold, O.: Synthesizers and their application to the parallel construction of pseudo-random functions. J. Comput. Syst. Sci. **58**(2), 336–375 (1999). https://doi.org/10.1006/JCSS.1998.1618
60. Page, A., Robert, D.: Introducing clapoti(s): Evaluating the isogeny class group action in polynomial time. Cryptology ePrint Archive, Report 2023/1766 (2023), https://eprint.iacr.org/2023/1766
61. Panny, L., Petit, C., Stopar, M.: KLaPoTi: An asymptotically efficient isogeny group action from 2-dimensional isogenies. Cryptology ePrint Archive, Paper 2024/1844 (2024), https://eprint.iacr.org/2024/1844
62. Peikert, C.: He gives C-sieves on the CSIDH. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part II. LNCS, vol. 12106, pp. 463–492. Springer, Cham (May 2020). https://doi.org/10.1007/978-3-030-45724-2_16
63. Petit, C., Lauter, K.: Hard and easy problems for supersingular isogeny graphs. Cryptology ePrint Archive, Report 2017/962 (2017), https://eprint.iacr.org/2017/962
64. Pizer, A.K.: Ramanujan graphs and Hecke operators. Bulletin of the American Mathematical Society **23**(1), 127–137 (1990)
65. Rabin, M.O.: Transaction protection by beacons. J. Comput. Syst. Sci. **27**(2), 256–267 (1983). https://doi.org/10.1016/0022-0000(83)90042-9

66. Unruh, D.: Post-quantum security of Fiat-Shamir. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part I. LNCS, vol. 10624, pp. 65–95. Springer, Cham (Dec 2017). https://doi.org/10.1007/978-3-319-70694-8_3
67. Xie, T., Zhang, Y., Song, D.: Orion: Zero knowledge proof with linear prover time. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part IV. LNCS, vol. 13510, pp. 299–328. Springer, Cham (Aug 2022). https://doi.org/10.1007/978-3-031-15985-5_11
68. Zeilberger, H., Chen, B., Fisch, B.: BaseFold: Efficient field-agnostic polynomial commitment schemes from foldable codes. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part X. LNCS, vol. 14929, pp. 138–169. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-68403-6_5

A Full R1CS description

Let $E_0 : y^2 = x^3 + A_0x^2 + C_0$ be the starting curve and $E_n : y^2 = x^3 + A_nx^2 + C_nx$ be the end curve of our CGL walk. We have the following witness, from which we omit the Re() and Im() parts for compactness,

$$z = (1, k_0, \dots, k_{n-1}, \beta_0, \dots, \beta_{n-1}, \alpha_0, \dots, \alpha_{n-1}, A_0, \dots, A_n, C_0, \dots, C_n, u_0, \dots, u_{n-1})\,.$$

Since $k_i, \beta_i, u_i \in \mathbb{F}_p$ and the $\alpha_i, A_i, C_i \in \mathbb{F}_p \times \mathbb{F}_p$, this encodes a total of $9n + 4$ variables over $\mathbb{F}_p$. At every step $E_i \to E_{i+1}$, we find the following set constraints, which immediately derive from equations (1)-(4).

$$\beta_i^2 = \mathrm{Re}(\alpha_i)$$

$$\begin{aligned} 2\mathrm{Re}(\alpha_i)\mathrm{Im}(\alpha_i) &= \mathrm{Im}(C_i) \\ 2(\mathrm{Re}(\alpha_i) + \mathrm{Im}(\alpha_i))(\mathrm{Re}(\alpha_i) + d\mathrm{Im}(\alpha_i)) &= 2\mathrm{Re}(C_i) + (d+1)\mathrm{Im}(C_i) \end{aligned}$$

$$\begin{aligned} 6k_i\mathrm{Re}(\alpha_i) &= \mathrm{Re}(A_{i+1}) - \mathrm{Re}(A_i) \\ 6k_i\mathrm{Im}(\alpha_i) &= \mathrm{Im}(A_{i+1}) - \mathrm{Im}(A_i) \end{aligned}$$

$$\begin{aligned} 2\mathrm{Im}(A_i)(\mathrm{Im}(A_{i+1}) - \mathrm{Im}(A_i)) &= u_i \\ 2\mathrm{Re}(A_i)(\mathrm{Re}(A_{i+1}) - \mathrm{Re}(A_i)) &= 3\mathrm{Re}(C_{i+1}) - 24\mathrm{Re}(C_i) - du_i \\ 2(\mathrm{Re}(A_i) + \mathrm{Im}(A_i))&(\mathrm{Re}(A_{i+1}) - \mathrm{Re}(A_i) + \mathrm{Im}(A_{i+1}) - \mathrm{Im}(A_i)) \\ &= 3\mathrm{Re}(C_{i+1}) - 24\mathrm{Re}(C_i) + 3\mathrm{Im}(C_{i+1}) - 24\mathrm{Im}(C_i) + (1-d)u_i \end{aligned}$$

Furthermore, we ensure that $k_i^2 = 1$ at every step. In the end, there are a total of $9n$ constraints for a path of length n.

We encode these constraints in the standard matrix description for R1CS. Since we are repeating the same constraints at every step, we only write down the submatrices for one step. Let

$$z_i = (1, k_i, \beta_i, \mathrm{Re}(\alpha_i), \mathrm{Im}(\alpha_i), \mathrm{Re}(A_i), \mathrm{Im}(A_i), \mathrm{Re}(C_i), \mathrm{Im}(C_i),$$

$$\mathrm{Re}(A_{i+1}), \mathrm{Im}(A_{i+1}), \mathrm{Re}(C_{i+1}), \mathrm{Im}(C_{i+1}), u_i)$$

be the subvector of z at step i, then the corresponding constraint equations are the following.

$$\begin{bmatrix} 0&0&1&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&2&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&2&2&0&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&2&0&0&0&0&0&0&0 \\ 0&0&0&0&0&2&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&2&2&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0&0&0&0&0 \end{bmatrix} z_i \circ \begin{bmatrix} 0&0&1&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&1&d&0&0&0&0&0&0&0&0&0 \\ 0&0&0&6&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&6&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&-1&0&0&0&1&0&0&0 \\ 0&0&0&0&0&-1&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&-1&-1&0&0&1&1&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0&0&0&0&0 \end{bmatrix} z_i$$

$$= \begin{bmatrix} 0&0&0&1&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&2&d+1&0&0&0&0&0 \\ 0&0&0&0&0&-1&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&-1&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&-24&0&0&0&3&0&-d \\ 0&0&0&0&0&0&0&-24&-24&0&0&3&3&0 \\ 1&0&0&0&0&0&0&0&0&0&0&0&0&0 \end{bmatrix} z_i$$

B Security Proofs

B.1 Security proofs for Section 4

We prove security of our VRF from Figure 2 below.

Theorem 4 (Provability). *The protocol from Figure 2 is provable if the proof system* NIZK *is correct and sound.*

Proof. If **KeyGen** and **Eval**$_k$ are correctly executed, then indeed $h = H(m, \pi_2)$, fulfilling the first verification condition. The second condition immediately follows from the correctness and soundness of NIZK. □

Theorem 5 (Unique Provability). *The protocol from Figure 2 is uniquely provable if* NIZK *is sound and* $\mathcal{W}(E_0, \cdot)$ *is injective, and weak uniquely provable if* $\mathcal{W}(E_0, \cdot)$ *is collision resistant.*

Proof. Suppose an adversary's two outputs $(h, (\pi, E))$ and $(h', (\pi', E'))$ are accepting for the same input m and public key pk. We will show that if NIZK is sound and $\mathcal{W}(E_0, \cdot)$ is injective (resp. collision resistant), then with overwhelming probability, $h = h'$. Let $\mathcal{L}$ be the language of relation $\mathcal{R}$ in (6).

Since the proofs are accepting and since NIZK is sound, it must be the case that both

$$\begin{aligned} h = H(E_k, m, E) \ \wedge \ (E_0, E_k, E_m, E) \in \mathcal{L} \,, \\ h' = H(E_k, m, E') \ \wedge \ (E_0, E_k, E_m, E') \in \mathcal{L} \,, \end{aligned}$$

up to negligible probability. This implies that both

$$\begin{aligned} \exists k : E_k = \mathcal{W}(E_0, k) \ \wedge \ E = \mathcal{W}(E_m, k) \,, \\ \exists k' : E_k = \mathcal{W}(E_0, k') \ \wedge \ E' = \mathcal{W}(E_m, k') \,. \end{aligned}$$

In the case where $\mathcal{W}(E_0, \cdot)$ is injective then it cannot be that $k \neq k'$. If $\mathcal{W}(E_0, \cdot)$ is collision resistant and $k \neq k'$, then the adversary has found a collision which occurs with most negligible probability. If $k = k'$, then $E = E'$ and finally $h = h'$.
□

Theorem 6 (Residual Pseudorandomness). *Let $\mathcal{A}_{RPR}$ be an adversary against the residual pseudorandomness game from Definition 6. Let further $\mathcal{B}_{ZK}$ be an adversary against the zero-knowledge property of $\mathsf{NIZK}_{\mathcal{L}}$ and $\mathcal{B}_{OME}$ an adversary against the one-more evaluation problem defined in Problem 4. If $\mathcal{A}$ is allowed up to q queries to the random oracle H and n queries to the $\mathbf{Eval}_{\mathsf{sk}}$ oracle, then*

$$\mathsf{Adv}(\mathcal{A}_{RPR}) \leq n\mathsf{Adv}(\mathcal{B}_{ZK}) + q\mathsf{Adv}(\mathcal{B}_{OME})$$

Proof. We prove the statement using four game hops. Let Game_0 be the residual pseudorandomness game from Definition 6 and let $\mathcal{A}$ be an adversary against Game_0 with advantage $\mathsf{Adv}(\mathcal{A}_{RPR})$. We define an adversary $\mathcal{B}_{ZK}$ against the zero-knowledge property of $\mathsf{NIZK}_{\mathcal{L}}$ and an adversary $\mathcal{B}_{OME}$ against the one-more evaluation problem defined in Problem 4. Let E_0 be a publicly available starting curve and let $(\mathsf{sk}, \mathsf{pk}) = (k, E_k)$ be the parameters of the one-more evaluation problem instance.

Game_1**:** Let $\mathcal{S}_{\mathsf{NIZK}}$ be the zero-knowledge simulator of $\mathsf{NIZK}_{\mathcal{L}}$, which on input x simulates a proof that $x \in \mathcal{L}$. We define Game_1 similar to Game_0, except that whenever $\mathcal{A}$ queries $\mathbf{Eval}_{\mathsf{sk}}$ on some input m, the reduction proceeds as follows.

1. Query $(h_0, (\pi_0, E)) \leftarrow \mathbf{Eval}_{\mathsf{sk}}(m)$,
2. compute $E_m = \mathcal{W}(E_0, m)$,
3. query $\pi_0' \leftarrow \mathsf{NIZK}_{\mathcal{L}}(E_0, E_k, E_m, E)$,
4. return $(h_0, (\pi_0', E))$ to $\mathcal{A}$.

If $\mathcal{A}$ can distinguish Game_1 from Game_0, then clearly $\mathcal{A}$ can be used to break the zero-knowledge property of $\mathsf{NIZK}_{\mathcal{L}}$. Assuming the number of $\mathcal{A}$'s queries to $\mathbf{Eval}_{\mathsf{sk}}$ is bounded by a parameter n, then the advantage of adversary $\mathcal{B}_{ZK}^{\mathcal{A}}$ against the zero-knowledge property of $\mathsf{NIZK}_{\mathcal{L}}$,

$$\mathsf{Adv}(\mathcal{B}_{ZK}) \geq \frac{1}{n} \big| \Pr(\mathcal{A} \text{ wins } \mathsf{Game}_1) - \Pr(\mathcal{A} \text{ wins } \mathsf{Game}_0) \big| \,.$$

Game_2**:** In this game, whenever $\mathcal{A}$ queries $\mathbf{Eval}_{\mathsf{sk}}$ on some input m, the adversary $\mathcal{B}_{OME}$, proceeds as follows.

1. Query $E \leftarrow \mathcal{O}_k(m)$,
2. query $h \leftarrow H(E_k, m, E)$,
3. query $\pi_0' \leftarrow \mathsf{NIZK}_{\mathcal{L}}(E_0, E_k, E_m, E)$,
4. return $(h_0, (\pi_0', E))$ to $\mathcal{A}$.

Since $E = \mathcal{W}(\mathcal{W}(E_0, m), k)$ by definition of the oracle $\mathcal{O}_k$, the output $(h_0, (\pi_0', E))$ is equal to Game_1 and therefore Game_2 is perfectly indistinguishable from Game_1.

Game_3**:** This game proceeds exactly as Game_2, except that $\mathcal{B}_{OME}$ simulates the random oracle, i.e. whenever $\mathcal{A}$ queries the random oracle H, instead $\mathcal{B}_{OME}$ samples $h_0' \leftarrow \{0,1\}^{2\lambda}$ and returns it to $\mathcal{A}$. $\mathcal{B}_{OME}$ keeps track of $\mathcal{A}$'s queries and returns the same output value for the same input value. Since $\mathcal{B}$ simulates the random oracle perfectly, this game is perfectly indistinguishable from Game_2.

Game_4**:** We define Game_4 in the same way as Game_3, with the exception that the random oracle H returns $\perp$ if it is queried on a previously defined critical input $x^* = (E_k, m^*, E^*)$, where $E^* = \mathcal{W}(\mathcal{W}(E_0, m^*), k)$.[7] If H is never queried on x^*, then Game_3 and Game_4 are perfectly indistinguishable and have the same success probability for the $\mathcal{A}$. Let X^* denote the event that H has been queried on x^* and $\neg X^*$ the event that it hasn't. We have

$$\Pr(\mathcal{A} \text{ wins } \mathsf{Game}_3 \mid \neg X^*) = \Pr(\mathcal{A} \text{ wins } \mathsf{Game}_4 \mid \neg X^*)\,. \tag{11}$$

In particular, this allows us to bound the difference in probability to solve either game as follows. Using the law of total probability, we find

$$\begin{aligned}&\big|\Pr(\mathcal{A} \text{ wins } \mathsf{Game}_3) - \Pr(\mathcal{A} \text{ wins } \mathsf{Game}_4)\big|\\ &\quad= \big|\big(\Pr(\mathcal{A} \text{ wins } \mathsf{Game}_3 \mid X^*) - \Pr(\mathcal{A} \text{ wins } \mathsf{Game}_4 \mid X^*)\big)\Pr(X^*) +\\ &\qquad \big(\Pr(\mathcal{A} \text{ wins } \mathsf{Game}_3 \mid \neg X^*) - \Pr(\mathcal{A} \text{ wins } \mathsf{Game}_4 \mid \neg X^*)\big)\Pr(\neg X^*)\big|\end{aligned}$$

Using (11), the second term vanishes and we can bound

$$\big|\Pr(\mathcal{A} \text{ wins } \mathsf{Game}_3) - \Pr(\mathcal{A} \text{ wins } \mathsf{Game}_4)\big| \leq \Pr(X^*)\,.$$

In a game where $\mathcal{A}$ cannot make the critical query x^* to the random oracle, the only way it can win Game_4 is by guessing the correct output, which implies that $Pr(\mathcal{A} \text{ wins } \mathsf{Game}_4) = \frac{1}{2}$, thus

$$\big|\Pr(\mathcal{A} \text{ wins } \mathsf{Game}_3) - \frac{1}{2}\big| \leq \Pr(X^*)\,.$$

The reduction. We now show how $\mathcal{B}_{OME}$ can turn an adversary $\mathcal{A}$ against Game_4 into an adversary against the one-more evaluation problem, which will we use to bound $\Pr(X^*)$. Using the reasoning from above, $\mathcal{A}$ must query

[7] This last step follows along the lines of the proof of [56, Proposition 2].

the random oracle on (E_k, m^*, E^*) in order to successfully distinguish the challenged strings h_0 and h_1. We therefore only need to consider the case where $\mathcal{A}$ does indeed send (E_k, m^*, E^*) as a query to the random oracle. We argue that submitting this query allows the algorithm $\mathcal{B}_{OME}$ to learn $E^* = \mathcal{W}(\mathcal{W}(E_0, m^*), k)$ and therefore break the one-more evaluation problem. Let the number of random oracle queries by $\mathcal{A}$ be at most q. $\mathcal{B}_{OME}$ proceeds as follows. (For simplicity, set $\mathcal{B} = \mathcal{B}_{OME}$ and $\mathcal{A} = \mathcal{A}_{RPR}$.)

1. $\mathcal{B}$ samples $i^* \leftarrow \{1, \dots, q\}$ and $h_0 \leftarrow \{0,1\}^{2\lambda}$.
2. $\mathcal{B}$ gets as input the public key $E_k = \mathcal{W}(E_0, k)$ and sends it to $\mathcal{A}$.
3. Whenever $\mathcal{A}$ sends its i-th query x_i to the random oracle,
   - if $i \neq i^*$ then $\mathcal{B}$ simulates the random oracle truthfully and keeps a list of $\mathcal{A}$'s queries and the related outputs,
   - if $i = i^*$ and x_{i^*} has been queried before, then $\mathcal{B}$ aborts, otherwise it sends h_0 to $\mathcal{A}$ and adds (x_{i^*}, h_0) to the random oracle list.
4. Whenever $\mathcal{A}$ sends a query m to $\mathbf{Eval}_k(\cdot)$, $\mathcal{B}$ proceeds as follows:
   (a) query $E \leftarrow \mathcal{O}_k(m)$,
   (b) query $h \leftarrow H(E_k, m, E)$,
   (c) use the zero-knowledge simulator $\mathcal{S}$ from $\mathsf{NIZK}_{\mathcal{L}}$ to build an accepting proof $\pi_1 \leftarrow \mathcal{S}((E_0, E_k, E_m, E))$,
   (d) send $(h, (\pi_1, E))$ to $\mathcal{A}$.
5. When $\mathcal{A}$ outputs m^*, $\mathcal{B}$ checks whether m^* has already been sent by $\mathcal{A}$ as a query to $\mathbf{Eval}_k(\cdot)$. In that case, $\mathcal{B}$ aborts and returns $\perp$. Otherwise $\mathcal{B}$ proceeds as follows:
   (a) sample $h_1 \leftarrow \{0,1\}^{2\lambda}$,
   (b) sample $b \leftarrow \{0,1\}$,
   (c) send h_b to $\mathcal{A}$.
6. For any further query to $H(\cdot)$ or $\mathbf{Eval}_k(\cdot)$, $\mathcal{B}$ proceeds as in steps 3 and 4, respectively.
7. At the end, $\mathcal{A}$ outputs b'.
8. Let $x_{i^*} = (E_k, m^*, E^*)$, then $\mathcal{B}$ returns (m^*, E^*).

Assuming that the query x_{i^*} was indeed (E_k, m^*, E^*) implies that $\mathcal{B}$ outputs a pair (m^*, E^*) that has not been submitted to $\mathcal{O}_k$, thus breaking the one-more evaluation problem. By randomly guessing one out of q queries by $\mathcal{A}$ to the random oracle, we find that $\mathsf{Adv}(\mathcal{B}_{OME}) \geq \frac{1}{q}\Pr(X^*)$.

Finally, we can compute the advantage of $\mathcal{A}$ against the original game Game_0 using the triangle inequality

$$\begin{aligned}
\mathsf{Adv}(\mathcal{A}_{RPR}) &= \big|\Pr(\mathcal{A} \text{ wins } \mathsf{Game}_0) - \frac{1}{2}\big| \\
&= \big|\Pr(\mathcal{A} \text{ wins } \mathsf{Game}_0) - \Pr(\mathcal{A} \text{ wins } \mathsf{Game}_4)\big| \\
&\leq \big|\Pr(\mathcal{A} \text{ wins } \mathsf{Game}_0) - \Pr(\mathcal{A} \text{ wins } \mathsf{Game}_1)\big| \\
&\qquad + \big|\Pr(\mathcal{A} \text{ wins } \mathsf{Game}_3) - \Pr(\mathcal{A} \text{ wins } \mathsf{Game}_4)\big| \\
&\leq n\mathsf{Adv}(\mathcal{B}_{ZK}) + q\mathsf{Adv}(\mathcal{B}_{OME}) \,.
\end{aligned}$$

□

B.2 Security proofs for Section 5

B.2.1 Proof of Theorem 2

Theorem 2. *The* CNIZK *proof system from Figure 3 is a non-interactive zero-knowledge proof of knowledge in the QROM.*

Proof. We argue correctness, special soundness and honest-verifier zero-knowledge of the underlying sigma protocol. Since the commitments are perfectly unpredictable and the responses $r_j(X)$ are perfectly unique for a given $f(X)$ and $b_j(X)$ (for positions $i = 0, 1, m$, this is guaranteed by the regularity of the group action), the results of the security of Fiat-Shamir in the QROM by [66] and [39] directly imply the truth of the theorem.

Correctness for $c_j = 0$ follows from $r_j(X) = b_j(X)$, while for $c_j = 1$ we have $r_j(X) + f(X) = b_j(X)$. For positions $i \in \{0, 1, m\}$, correctness follows from the fact that $[r_j(i)][f(i)]E = [r_j(i) + f(i)]E = [b_j(i)]E$.

Assume for the different challenges $c_j = 0$ and $c_j = 1$, we have the accepting responses $r_j(X)$ and $r'_j(X)$. From the verification conditions, it follows that for $i = 1, 2, m$, $[r_j(i)]E_0 = [r'_j(i)]\mathsf{P}_i$ and therefore $[r_j(i) - r'_j(i)]E_0 = \mathsf{P}_i$ (for simplicity, let $\mathsf{P}_m = E_m$). Thus $r_j(i) - r'_j(i)$ allow to recover valid witnesses $f(i)$ for $i = 0, 1, m$, which are witnesses for the key recovery problem (Problem 2). Note that given the public evaluations $f(2), \dots, f(d)$, one can further recover the entire polynomial $f(X)$, which is a witness for relation (10).

A simulator with input (pk, m, E_m) can generate a transcript of the protocol by sampling $c_j \leftarrow \{0, 1\}$ and $r_j(X) \leftarrow \mathbb{Z}_N[X]$ of degree d for $j = 1, \dots, \lambda$, then setting $S_j = ([r_j(0)]E_0, [r_j(1)]E_0, r_j(2), \dots, r_j(d), [r_j(m)]E_0)$ if $c_j = 0$ and $S_j = ([r_j(0)]P_0, [r_j(1)]P_1, r_j(2) + f(2), \dots, r_j(d) + f(d), [r_j(m)]E_m)$ if $c_j = 1$, and reprogramming the random oracle to yield the desired result. □

B.2.2 Proof of Theorem 3

We prove the following three theorems.

Theorem 7 (Provability). *The protocol from Figure 4 is provable if* CNIZK *is correct and sound.*

Proof. If $\mathbf{KeyGen}_d(1^\lambda)$ and $\mathbf{Eval}_f(m)$ are correctly evaluated, then indeed $E_m = [f(m)]E_0$. Correctness and soundness of CNIZK imply that π is indeed a proof of the statement in equation (10), except when the prover has cheated, which only happens with negligible probability. □

Theorem 8 (Unique Provability). *The protocol from Figure 4 is uniquely provable if* CNIZK *is correct and sound.*

Proof. Assume we have two accepting outputs $h = H(E_k, m, E_m)$ and $h' = H(E_k, m, E'_m)$. Since H is modeled as a random oracle, $h \neq h'$ implies $E_m \neq E'_m$, except for a collision, which occurs only with negligible probability. Since both outputs are accepting, however, there must exist $f(X), f'(X) \in \mathbb{Z}_N[X]$ of degree $\deg f, f' \leq d$ such that $E_m = [f(m)]E_0$ and $E'_m = [f'(m)]E_0$. Since $f(i) = f'(i)$ for $i = 0, \dots, d$ in the public key, it must be that $f(X) = f'(X)$, which implies $E_m = E'_m$, and therefore $h = h'$. □

Theorem 9 (Residual Pseudorandomness). *Let $\mathcal{A}_{RPR}$ be an adversary against the residual pseudorandomness game from Definition 6. Let further $\mathcal{B}_{ZK}$ be an adversary against the zero-knowledge property of* CNIZK *and $\mathcal{B}_{OMU}$ an adversary against the one-more unpredictability problem defined in Problem 4. If $\mathcal{A}$ is allowed up to q queries to the random oracle H and n queries to the* $\mathbf{Eval}_{\mathsf{sk}}$ *oracle, then*

$$\mathsf{Adv}(\mathcal{A}_{RPR}) \leq n\mathsf{Adv}(\mathcal{B}_{ZK}) + q\mathsf{Adv}(\mathcal{B}_{OMU})$$

Proof. The proof works completely analogous to the proof of Theorem 6 by replacing Problem 4 with Problem 5 and $\mathsf{NIZK}_{\mathcal{L}}$ with CNIZK. We do the same game hops as in the proof of Theorem 6, incrementally replacing the zero-knowledge proofs with the CNIZK-simulator, then the $\mathbf{Eval}_{\mathsf{sk}}$-oracle with $\mathsf{VRF}_f(\cdot)$ from Problem 5 and simulated proofs and finally simulating the random oracle. We have shown that a successful adversary must have queried the random oracle on $(\mathsf{pk}, m^*, E_{m^*})$ at some point, except if it simply guessed E_{m^*}, which happens only with negligible probability. We can then define an adversary $\mathcal{B}_{OMU}$ against Problem 5 that runs $\mathcal{A}$ as a subroutine exactly in the same way as in the proof of Theorem 6, but with respect to Figures 3 and 4. We have shown in the proof of Theorem 6 that this can be done in a way that for $\mathcal{A}$, this is indistinguishable from the real execution of the protocol. If out of $\mathcal{A}$'s q queries to the random oracle, $\mathcal{B}_{OMU}$ guessed the correct one, then it can output a tuple $(m^*, [f(m^*)]E_0)$, which has not been queried to $\mathbf{Eval}_f(\cdot)$ before, thus breaking Problem 5. Assuming up to n queries to the $\mathbf{Eval}_f(\cdot)$, we find the advantage described in the theorem. □