Triple Cryptanalysis of Isogeny-Based VRFs from Asiacrypt 2025

Yi-Fu Lai[1], Yu Yu[1], and Xiaogang Zhou[2]

[1] Shanghai Jiao Tong University
[2] China Telecom Quantum Information Technology Group Co., Ltd

Abstract. Levin and Pedersen proposed at Asiacrypt2025 a new verifiable random function (VRF) based on a CGL-analogue hash function constructed from radical isogenies. Their construction applies the same secret radical-CGL walk to a public starting curve and a message-dependent curve, and uses an R1CS proof relation to show that the two walks use the same secret key.

We present a two-stage attack on this construction. The first stage concerns the unspecified representation of the public key. The reported key size indicates that the public curve is stored as a j-invariant, whereas both the specified radical-CGL computation use two coefficients to represent a curve. By exploiting this form we can produce two different VRF outputs under the same public key and message, breaking the unique provability. Hence, the output of the radical-CGL computation must follow the specification.

In the second stage, we exploit these coefficients to recover the VRF secret key. With 1536 queries, our implementation recovers the complete 256-bit secret in 30 minutes, thereby breaking residual pseudorandomness. Interestingly, we also observe that the using public key alone without queries can sometimes reveal one or two bits of the secret walk.

Besides, we extend Lai's observation to obtain a one-query attack on the group-action-based VRF proposed in the same paper with advantage closed to 1/2. Together, these constitute three attacks on their work.

1 Introduction

A verifiable random function (VRF) is a keyed deterministic function whose output is accompanied by a publicly verifiable proof. Besides correctness, a VRF requires unique provability and residual pseudorandomness: for a fixed verification key and message, two different outputs should not both verify, and the output on a fresh message should remain indistinguishable from random even after seeing evaluations on other messages [12].

Levin and Pedersen proposed at Asiacrypt2025 a post-quantum VRF from CGL walks in the supersingular 2-isogeny graph [11]. Their construction uses the radical degree-2 isogeny formula of Castryck, Decru, and Vercauteren [2]. Let E_0 be a public supersingular starting curve and let $\overrightarrow{k} \in \{\pm 1\}^n$ be the secret walk. On a message $\overrightarrow{m} \in \{\pm 1\}^n$, the evaluator computes $E_{\overrightarrow{m}} = \mathsf{CGL}(E_0, \overrightarrow{m})$ and

then reuses the same secret walk from this message-dependent curve to obtain $E_{\overrightarrow{m},\overrightarrow{k}} = \mathsf{CGL}(E_{\overrightarrow{m}}, \overrightarrow{k})$. The verification key is $E_{\overrightarrow{k}} = \mathsf{CGL}(E_0, \overrightarrow{k})$. A same-input NIZK proves that the two paths $E_0 \to E_{\overrightarrow{k}}$ and $E_{\overrightarrow{m}} \to E_{\overrightarrow{m},\overrightarrow{k}}$ use the same walk $\overrightarrow{k}$, and the final VRF value hashes the verification key, message, and evaluation curve. For NIST level I, the proposal takes $p = 5 \cdot 2^{248} - 1$ and $n = 256$. It estimates a fast 85 ms evaluation, a 30 ms verification, and an output of about 430 kB [11], which is dominated by proof generated by Aurora and can be replaced by any other generic tool.

The original security analysis argues that the design makes correlations across different evaluations implausible. First, the scheme uses a starting curve with unknown endomorphism ring to preclude known collision attacks based on short cycles. Second, although the same bit string k is reused for walks starting from different curves E_m, the two radical roots are reordered independently at every intermediate state. Specifically, [11] defines the canonical root according to the quadratic residuosity of its real component, or of its imaginary component when the real component vanishes. The authors argue that this curve-dependent ordering prevents the bits of k from retaining a consistent algebraic interpretation across different starting curves. They further support this claim experimentally by comparing output distributions and graph distances, using bounded-distance Dijkstra search to measure the distances between resulting walks [11, Section 4.2.1 and Appendix C].

1.1 Contributions

Our contributions are summarized as follows.

- **Failure of weak unique provability.** The generic construction of [11, Theorem 1] requires the underlying map to be injective. If the public key output by the radical-CGL hash is stored only as a j-invariant, without additional information fixing its coefficient representation, then we show it will violate the premise of the security theorem and invalidate the proofs. Moreover, we give a concrete adversary that produces two accepting evaluations with distinct VRF outputs for the same message and public key, represented by a j-invariant, thereby breaking weak unique provability; see Section 3.2.
- **Full secret-key recovery.** We next follow the well-defined coefficient-pair evaluation used by the Radical-CGL algorithm in the original proposal. We propose a novel algebraic attack. The attack exploits the representations of elliptic curves to recover the hidden algebraic structure between curves. This yields a new key-recovery attack against the complete secret walk. For the proposed 256-bit secret key, our implementation uniquely recovers all 256 bits using 1536 evaluation queries; see Sections 3.3 and 5.
- **Partial key leakage.** Interestingly, the structure can reveal the one or two bits of the secret walk from a single coefficient-pair endpoint, without any queries. Similar obeservation can apply to classic CGL hash using j-invairants [4,5]. This does not make the full key recovery easier. We discuss this additional leakage in Section 3.

– **One-query attack on the polynomial group-action VRF.** We extend Lai's observation on the polynomial group-action construction [10] and show a single evaluation query will restrict the evaluation at a small, publicly enumerable subgroup orbit by refining the technique proposed in [8]. For the proposed group order $N = 2q$, we can break the residual pseudorandomness with advantage around $1/2$; see Section 4.1.

2 Preliminaries

Notation. Throughout the paper, λ denotes the security parameter and $\mathsf{negl}(\lambda)$ denotes a negligible function in λ. For a finite set S, $x \xleftarrow{\$} S$ denotes uniform sampling from S. We use $\overrightarrow{x}$ for a vector or sequence and $u \,\|\, v$ for concatenation. All adversaries are probabilistic polynomial-time unless stated otherwise.

2.1 Verifiable random functions

A verifiable random function is a tuple of polynomial-time algorithms $\mathsf{VRF} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Verify})$. On input 1^λ, Setup returns public parameters pp. The algorithm $\mathsf{KeyGen}(pp)$ returns (sk, pk); $\mathsf{Eval}_{sk}(m)$ returns a value and proof (h, π); and $\mathsf{Verify}_{pk}(m, h, \pi)$ returns a bit. We recall the three properties used in the target work [11, Section 2.3] which is the same as [12].

Definition 1 (Provability). *For every message m, sample $pp \leftarrow \mathsf{Setup}(1^\lambda)$, $(sk, pk) \leftarrow \mathsf{KeyGen}(pp)$, and $(h, \pi) \leftarrow \mathsf{Eval}_{sk}(m)$. Provability requires*

$$\Pr[\mathsf{Verify}_{pk}(m, h, \pi) = 1] \geq 1 - \mathsf{negl}(\lambda).$$

Definition 2 (Weak unique provability). *The experiment samples $pp \leftarrow \mathsf{Setup}(1^\lambda)$ and obtains $(pk, m, h_0, \pi_0, h_1, \pi_1) \leftarrow \mathcal{A}(1^\lambda, pp)$. The adversary wins if*

$$\mathsf{Verify}_{pk}(m, h_0, \pi_0) = \mathsf{Verify}_{pk}(m, h_1, \pi_1) = 1 \quad \text{and} \quad h_0 \neq h_1. \tag{1}$$

Weak unique provability, roughly speaking, requires that for a PPT adversary it is difficult to find two valid evaluations for the same message under the same verification key.

Definition 3 (Residual pseudorandomness). *Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be PPT. The challenger generates $pp \leftarrow \mathsf{Setup}(1^\lambda)$ and $(sk, pk) \leftarrow \mathsf{KeyGen}(pp)$. With access to $\mathsf{Eval}_{sk}(\cdot)$ and the random oracle, $\mathcal{A}_1(pk)$ returns a fresh challenge message m^* and state st. The challenger computes $(h_0, \pi_0) \leftarrow \mathsf{Eval}_{sk}(m^*)$, samples $h_1 \xleftarrow{\$} \{0,1\}^{2\lambda}$ and $b \xleftarrow{\$} \{0,1\}$, and gives (pk, h_b, st) to $\mathcal{A}_2$, which retains the same oracle access. Let b' be the output of $\mathcal{A}_2$. The adversary wins when $b' = b$ and it never queried $\mathsf{Eval}_{sk}(m^*)$. Residual pseudorandomness requires its advantage over $1/2$ to be negligible.*

2.2 Elliptic curves

Let K be a field of characteristic different from 2 and 3. Two elliptic curves over an algebraic closure of K are isomorphic if and only if they have the same j-invariant. We use the model

$$E_{A,C} : y^2 = x^3 + Ax^2 + Cx, \tag{2}$$

where $A, C \in K$ and $C(A^2 - 4C) \neq 0$. As shown in [2,11], the point $(0, 0)$ is a nonzero 2-torsion point, and

$$j(E_{A,C}) := 256\frac{(A^2 - 3C)^3}{C^2(A^2 - 4C)}. \tag{3}$$

For convenience, we denote $j(E_{A,C})$ as $j(A, C)$.

2.3 Radical isogenies and CGL walks

Let $p \equiv 3 \pmod 4$ and work over $\mathbb{F}_{p^2}$. Fix a public deterministic square-root selector rt on the nonzero squares of $\mathbb{F}_{p^2}$, with $\mathsf{rt}(C)^2 = C$. Levin and Pedersen select the root whose real component is a quadratic residue in $\mathbb{F}_p$, using the imaginary component when the real component is zero [11, Section 3.1]. The specific choice of rt is irrelavant to our attack. In [11], $b \in \{\pm 1\}$, one radical-CGL step is

$$T_b(A, C) = \big(A + 6b \cdot \mathsf{rt}(C),\ 8C + 4b \cdot \mathsf{rt}(C)A\big). \tag{4}$$

We also call this b as a label. This is the iterative degree-2 radical formula obtained by translating a suitable nonzero 2-torsion point to $(0, 0)$ For $w = (w_1, \ldots, w_d) \in \{\pm 1\}^d$, write $\mathsf{CGL}(E, w) = T_{w_d} \circ \cdots \circ T_{w_1}(E)$. As defined in [11, Algorithm 1], the authors use two coefficients to represent the output.

2.4 The radical-CGL VRF

The work [11] defines the VRF with CGL by considering the following same-input relation:

$$\mathcal{R}^{\mathsf{parallel}}_{\mathsf{CGL}} = \left\{((E_1, E_2), (F_1, F_2); \overrightarrow{k}) : E_2 = \mathsf{CGL}(E_1, \overrightarrow{k}) \ \wedge \ F_2 = \mathsf{CGL}(F_1, \overrightarrow{k})\right\}. \tag{5}$$

Concretely, the VRF evaluation by computing Eq. (4) twice and shares the label variables $k_1, \ldots, k_n$ between the two paths: from E_1 to E_2 and from F_1 to F_2.

$\mathsf{Setup}(1^\lambda)$

1 : $pp \leftarrow (\mathsf{CGL}, \mathcal{E}, \{\pm 1\}^n, E_0, \mathsf{NIZK}, \mathsf{H})$
2 : **return** pp

$\mathsf{KeyGen}(pp)$

1 : $\overrightarrow{k} \xleftarrow{\$} \{\pm 1\}^n$
2 : $E_k \leftarrow \mathsf{CGL}(E_0, \overrightarrow{k})$
3 : **return** $(sk, pk) = (\overrightarrow{k}, E_{\overrightarrow{k}})$

$\mathsf{Eval}_{\overrightarrow{k}}(\overrightarrow{m})$

1 : $E_m \leftarrow \mathsf{CGL}(E_0, \overrightarrow{m})$
2 : $E_{\mathsf{eval}} \leftarrow \mathsf{CGL}(E_{\overrightarrow{m}}, \overrightarrow{k})$
3 : $\pi_1 \leftarrow \mathsf{NIZK.P}(\overrightarrow{k}; (E_0, E_{\overrightarrow{k}}, E_{\overrightarrow{m}}, E_{\mathsf{eval}}))$
4 : $h \leftarrow \mathsf{H}(E_{\overrightarrow{k}}, \overrightarrow{m}, E_{\mathsf{eval}})$
5 : **return** $(h, (\pi_1, E_{\mathsf{eval}}))$

$\mathsf{Verify}_{E_k}(m, h, (\pi_1, E_{\mathsf{eval}}))$

1 : $E_m \leftarrow \mathsf{CGL}(E_0, \overrightarrow{m})$
2 : $b_0 \leftarrow [h = \mathsf{H}(E_k, \overrightarrow{m}, E_{\mathsf{eval}})]$
3 : $b_1 \leftarrow \mathsf{NIZK.V}(\pi_1; (E_0, E_{\overrightarrow{k}}, E_{\overrightarrow{m}}, E_{\mathsf{eval}}))$
4 : **return** $b_0 \wedge b_1$

Fig. 1. The radical-CGL VRF in the abstract curve syntax of [11, Figure 1] with $\mathsf{NIZK} = \{\mathsf{NIZK.P}, \mathsf{NIZK.V}\}$. E_0 is a trusted setup curve with unknown endomorphism ring.

For level I, the proposal sets $n = 256$ and $p = 5 \cdot 2^{248} - 1$. In light of Wesolowski's attack [13], our implementation retains $n = 256$ but uses the SQIsign Round-2 Level-3 prime, showing that our attack does not rely on the original field size.

The proposal estimates the public-key size of the radical-CGL VRF as 64 bytes but does not specify its concrete encoding [11]. For the proposed parameters, a single element of $\mathbb{F}_{p^2}$ occupies approximately 64 bytes, whereas the coefficient pair $(A, C) \in \mathbb{F}_{p^2}^2$ required by the radical-CGL recurrence occupies approximately 128 bytes. Since the distribution experiment in [11, Fig. 4] represents the output of $\mathsf{CGL}(m, k)$ by its j-invariant, we therefore interpret the advertised 64-byte public key as the compact value $j(E_{\overrightarrow{k}})$. This representation, however, is insufficient for the coefficient-level computation: the radical-CGL algorithm and the R1CS relation require an explicit pair (A, C) representing the public endpoint. The proposal specifies neither a unique lift from $j(E_{\overrightarrow{k}})$ to such a pair nor constraints that uniquely bind the pair used in the proof to the public j-invariant. We will show this missing interface is security-critical: first in theory, it will contradict to the injectivity required in [11, Theorem 1]. Second, we will show allowing different coefficient representatives of the same public j-invariant violates the unique provability.

2.5 Effective group actions and a VRF construction

Let $(G, +)$ be a finite abelian group, let $\mathcal{E}$ be a finite set, and let

$$[\cdot] \cdot : G \times \mathcal{E} \longrightarrow \mathcal{E}$$

be a regular group action. We further need the following basic properties [1,7] for the cryptographic purpose.

1. **Group-action laws.** For every $a, b \in G$ and $E \in \mathcal{E}$,
$$[0]\,E = E, \qquad [a+b]\,E = [a]\,\big([b]\,E\big).$$
2. **Regularity.** For every $E, E' \in \mathcal{E}$, there exists a unique $a \in G$ such that $E' = [a]\,E$.
3. **Effectiveness.** An effective group action additionally provides efficient algorithms for group operations in G, equality and membership testing in G and $\mathcal{E}$, and evaluation of $[a]\,E$.
4. **Representation.** There are unique representations for the group and the set elemenet, which can be computed effectively.

The target work assumes a known cyclic group $G \cong \mathbb{Z}_N$ acting regularly and effectively on $\mathcal{E}$, together with a public origin $E_0 \in \mathcal{E}$ [3,11].

Polynomial group-action VRF of Levin and Pedersen. The second VRF of Levin and Pedersen [11, Section 5] is defined over a regular effective group action $(\mathbb{Z}_N, \mathcal{E}, E_0)$. For a fixed degree $d \in \{1,2,3\}$, the secret key is a polynomial $f(X) \in \mathbb{Z}_N[X]$ of degree d. Writing $P_i = [f(i)]E_0$, the public key is

$$pk = \begin{cases} (P_0, f(1)), & d = 1, \\ (P_0, P_1, f(2), \ldots, f(d)), & d \geq 2. \end{cases} \tag{6}$$

The construction follows the VRF syntax $\mathsf{VRF} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Verify})$ recalled in Section 2.1.

The setup algorithm $\mathsf{Setup}(1^\lambda)$ outputs public parameters pp describing $(\mathbb{Z}_N, \mathcal{E}, E_0)$, the hash function H, and the proof system NIZK. The key-generation algorithm $\mathsf{KeyGen}_d(pp)$ samples $f(X) \in \mathbb{Z}_N[X]$ of degree d with nonzero coefficients and returns

$$(sk, pk) = \begin{cases} \big(f(X), (P_0, f(1))\big), & d = 1, \\ \big(f(X), (P_0, P_1, f(2), \ldots, f(d))\big), & d \geq 2. \end{cases}$$

On input $m \in \mathbb{Z}_N$, the evaluation algorithm computes

$$E_m \leftarrow [f(m)]E_0, \qquad \pi \leftarrow \mathsf{NIZK.P}\big(f(X); (pk, m, E_m)\big), \qquad h \leftarrow \mathsf{H}(pk, m, E_m),$$

and returns $\mathsf{Eval}_f(m) = \big(h, (E_m, \pi)\big)$. Here, h is the VRF value, while (E_m, π) is the VRF proof. In particular, E_m gives the underlying group-action evaluation, and π proves that there exists a polynomial $f(X) \in \mathbb{Z}_N[X]$ of degree at most d such that pk has the form in Eq. (6) and $E_m = [f(m)]E_0$.

Finally, the verification algorithm computes

$$\mathsf{Verify}_{pk}\big(m, h, (E_m, \pi)\big) = \big[h = \mathsf{H}(pk, m, E_m)\big] \wedge \mathsf{NIZK.V}\big(\pi; (pk, m, E_m)\big). \tag{7}$$

Residual pseudorandomness is reduced in [11] to the one-more unpredictability problem introduced in [6]: given oracle access to $[f(m)]E_0 \leftarrow \mathcal{O}_f(m)$, output a pair $\big(m^*, [f(m^*)]E_0\big)$ for an input m^* that was not previously queried to $\mathcal{O}_f$.

3 Cryptanalysis on Radical-CGL-based VRF

3.1 Attack Strategy

Our attack has two stages. First we show that there must be a canonical way compute the coefficients A, C for a curve from a j-invariant public key; otherwise, the VRF evaluation is not well-defined. In other words, we can break the unique provability. Based on the previous requirement, we show a novel key recovery attack, which breaks the pseudorandomness. In our experiment, the full 256-bit secret key can be recovered with around 1500 queries only within 30 minutes.

3.2 First stage: failure of weak unique provability

First observe that a j-invariant of curve admit multiple coefficients in (A, C).

Lemma 1. *Let $E_{A,C}$ be as in Eq. (2). For every $\lambda \in K^\times$, one has $j(\lambda A, \lambda^2 C) = j(A, C)$. If $\lambda = u^{-2}$ for some $u \in K^\times$, then $E_{A,C}$ and $E_{\lambda A, \lambda^2 C}$ are K-isomorphic.*

If $r \in K$ satisfies $r^2 + Ar + C = 0$, then translating $(r, 0)$ to $(0, 0)$ gives the K-isomorphic model

$$y^2 = x^3 + (A + 3r)x^2 + r(A + 2r)x. \tag{8}$$

Proof. The first identity follows by substituting $(\lambda A, \lambda^2 C)$ into Eq. (3). The change of variables $x = u^2 x'$ and $y = u^3 y'$ gives the coefficients $u^{-2}A$ and $u^{-4}C$. Finally, substituting $x = x' + r$ into Eq. (2) gives Eq. (8); its constant term is zero because $r^2 + Ar + C = 0$.

Although an elliptic-curve isomorphism class admits many coefficient representations, our analysis target is very concrete. In a modular-polynomial realization of the classic CGL hash [4], distinct choice at each step will walk to a distinct curve with respect to j-invariants.

The degree-2 radical recurrence behaves differently. Two choices at each step will result in the isomorphic curves to each other in the different representations. This is becasue each step takes the distinguished subgroup $\langle (0, 0) \rangle$ as the kernel. The two square-root choices, up to the secret bit, do not define two different quotients; instead, they select one of the two nonzero 2-torsion points on the codomain outside the kernel of the dual isogeny. The selected point is then translated to $(0, 0)$ so that it becomes the distinguished kernel for the next iteration. Consequently, the two resulting coefficient pairs are isomorphic models of the same quotient curve and have the same j-invariant, while encoding different choices for the next non-backtracking step. Alternatively, one can prove this in pure algebraic computation as the following lemma.

Lemma 2 (Terminal-branch collision). *Let (A, C) be a valid radical state, let $\alpha^2 = C$, and set $(A_b, C_b) = (A + 6b\alpha, 8C + 4b\alpha A)$ for $b \in \{\pm 1\}$. Then*

$$j(A_{+1}, C_{+1}) = j(A_{-1}, C_{-1}) = 16\frac{(A^2 + 12C)^3}{C(A^2 - 4C)^2}. \tag{9}$$

Proof. For either $b \in \{\pm 1\}$, one has $A_b^2 - 3C_b = A^2 + 12C$, $A_b^2 - 4C_b = (A - 2b\alpha)^2$, and $C_b = 4\alpha(2\alpha + bA)$. Hence $C_b^2(A_b^2 - 4C_b) = 16C(A^2 - 4C)^2$. Substitution into Eq. (3) proves the claim. □

Under the compact-key issue mentioned in Section 2.3, the public key is

$$pk = j(E_{\overrightarrow{k}}), \qquad E_{\overrightarrow{k}} = \mathsf{CGL}(E_0, \overrightarrow{k}),$$

whereas the same-input relation is defined over explicit coefficient states. Consequently, to instantiate the published coefficient-level relation, the prover must provide a representative $K = (A_K, C_K)$ of the public curve. Since the proposal specifies neither a canonical lift of pk nor any additional condition that uniquely determines K, the verifier can only bind this representative to the public key by checking $j(K) = pk$.

Concretely, the evaluation proof contains $\Pi = (K, E, \pi_1)$, where K and E are the coefficient states used in the two instances of the radical-CGL relation. The corresponding verifier accepts if

$$j(K) = pk, \qquad h = \mathsf{H}(pk, \overrightarrow{m}, E), \qquad \mathsf{NIZK.V}\big(\pi_1; (E_0, K, E_{\overrightarrow{m}}, E)\big) = 1, \quad (10)$$

where $E_{\overrightarrow{m}} = \mathsf{CGL}(E_0, \overrightarrow{m})$. Thus, although pk fixes an isomorphism class, it does not uniquely fix the coefficient representative appearing in the proof statement. The terminal-branch collision below exploits precisely this ambiguity to produce two accepting evaluations for the same public key and message.

$\mathsf{BreakWeakUniqueness}(pp)$

$u \xleftarrow{\$} \{\pm 1\}^{n-1}$; $\quad \overrightarrow{m} \xleftarrow{\$} \{\pm 1\}^n$

$\overrightarrow{k_+} \leftarrow u \,\|\, (+1)$; $\quad \overrightarrow{k_-} \leftarrow u \,\|\, (-1)$

$K_+ \leftarrow \mathsf{CGL}(E_0, \overrightarrow{k_+})$; $\quad K_- \leftarrow \mathsf{CGL}(E_0, \overrightarrow{k_-})$

$pk \leftarrow j(K_+)$

$E_{\overrightarrow{m}} \leftarrow \mathsf{CGL}(E_0, \overrightarrow{m})$

$E_+ \leftarrow \mathsf{CGL}(E_m, \overrightarrow{k_+})$; $\quad E_- \leftarrow \mathsf{CGL}(E_m, \overrightarrow{k_-})$

$\pi_+ \leftarrow \mathsf{NIZK.P}(\overrightarrow{k_+}; (E_0, K_+, E_m, E_+))$

$\pi_- \leftarrow \mathsf{NIZK.P}(\overrightarrow{k_-}; (E_0, K_-, E_m, E_-))$

$h_+ \leftarrow \mathsf{H}(pk, \overrightarrow{m}, E_+)$; $\quad h_- \leftarrow \mathsf{H}(pk, \overrightarrow{m}, E_-)$

return $(pk, \overrightarrow{m}, h_+, (K_+, E_+, \pi_+), h_-, (K_-, E_-, \pi_-))$

Fig. 2. Weak-unique-provability attack on the non-canonical coefficient interface.

Theorem 1. *Consider the compact-key interpretation of the radical-CGL VRF with the verifier in Eq. (10). If* H *is modeled as a random oracle with range*

$\{0,1\}^{2\lambda}$, *then the adversary in Figure 2 violates weak unique provability with probability at least*

$$1 - 2^{-2\lambda} - \mathsf{negl}(\lambda).$$

Proof. Let $P = \mathsf{CGL}(E_0, u)$ and $M = \mathsf{CGL}(E_{\overrightarrow{m}}, u)$. The two public-key states are obtained by applying the two terminal labels to the same state P: $K_+ = T_{+1}(P)$ and $K_- = T_{-1}(P)$. By Lemma 2, $j(K_+) = j(K_-)$, so both coefficient representatives are accepted under the same compact verification key

$$pk = j(K_+) = j(K_-).$$

Similarly, the two evaluation states satisfy

$$E_+ = T_{+1}(M) \qquad \text{and} \qquad E_- = T_{-1}(M).$$

Writing $M = (A, C)$, their first coefficients are $A + 6\mathsf{rt}(C)$ and $A - 6\mathsf{rt}(C)$, respectively. Since M is a valid radical state, $C \neq 0$, and hence $\mathsf{rt}(C) \neq 0$. Therefore $E_+ \neq E_-$.

For each sign $b \in \{+1, -1\}$, the witness $\overrightarrow{k_b}$ satisfies

$$K_b = \mathsf{CGL}(E_0, \overrightarrow{k_b}) \qquad \text{and} \qquad E_b = \mathsf{CGL}(E_{\overrightarrow{m}}, \overrightarrow{k_b}).$$

Moreover, $j(K_b) = pk$, so both tuples satisfy all checks in Eq. (10).

Finally, since $E_+ \neq E_-$, the random-oracle inputs $(pk, \overrightarrow{m}, E_+)$ and $(pk, \overrightarrow{m}, E_-)$ are distinct except for the negligible chance. Consequently,

$$\Pr[h_+ = h_-] = 2^{-2\lambda} + \mathsf{negl}(\lambda).$$

Except for this collision event and the negligible completeness error, the adversary outputs two accepting evaluations with $h_+ \neq h_-$ for the same public key and message, and therefore wins the experiment in Definition 2.

To prevent the attack in Theorem 1 while maintaining the coefficient-level relation in Eq. (5), the scheme must uniquely bind each public curve representation to the coefficient state used by the R1CS. This can be achieved, for example, by publishing the coefficient pair directly or by prescribing a public deterministic lift from the compact curve representation. By following the description of [11, Algorithm 1], we opt for the former option where the public key and the evaluation curves are available as coefficient pairs in $(\mathbb{F}_{p^2})^2$. Throughout the rest of paper, we analyze the scheme in this form.

3.3 Second stage: exact inversion of one radical step

Now we may assume that the coefficient representation of each public curve, including each evaluation curve, is uniquely determined. The subsection presents the key recovery attack. Starting from an evaluation curve, the natural strategy is then to reverse the secret walk one step at a time.

Given E and $\widetilde{E}$, we call E a *predecessor* of $\widetilde{E}$ if the recurrence formula maps E to $\widetilde{E}$. Recall that one forward step maps $E_{A,C}$ to $\widetilde{E} = E_{\widetilde{A},\widetilde{C}}$ by

$$\widetilde{A} = A + 6b \cdot \mathsf{rt}(C), \qquad \widetilde{C} = 8C + 4b \cdot \mathsf{rt}(C)A, \qquad b \in \{\pm 1\}.$$

This recurrence can be inverted algebraically. Set $\delta = \widetilde{A} - A$. The first coordinate gives $\delta = 6b \cdot \mathsf{rt}(C)$ and hence $C = \delta^2/36$. Substituting $A = \widetilde{A} - \delta$ and $C = \delta^2/36$ into the second coordinate gives $4\delta^2 - 6\widetilde{A}\delta + 9\widetilde{C} = 0$. Thus, writing $\beta^2 = \widetilde{A}^2 - 4\widetilde{C}$, one obtains two possible predecessors

$$E_{\pm} = E_{A_{\pm},C_{\pm}}, \qquad A_{\pm} = \frac{\widetilde{A} \mp 3\beta}{4}, \qquad C_{\pm} = \left(\frac{\widetilde{A} \pm \beta}{8}\right)^2.$$

At first sight, this two equivocal candidates seem to make backtracking useless. The actual previous curve is among these two possibilities, but the inverse formula does not tell us which one belongs to the secret walk. The advantage is the same as guessing the secret bit.

The key observation is that the two candidates do not necessarily return to $\widetilde{E}$ *under opposite labels*. Since the canonical square-root operation is evaluated independently on the two different squares C_+ and C_-. Consequently, $s_+ = -s_-$ will not always hold. The two candidates may instead have label pattern $++$, $+-$, or $--$. Critically when the pattern has the repeated sign, we are able to recover the last label (i.e. the last secret bit).

We now formalize these notions.

Definition 4 (Predecessors). *Let $\widetilde{E}$ be a curve within the radical walk and let $b \in \{\pm 1\}$. A curve E is called a* b-labeled predecessor *of $\widetilde{E}$ if $T_b(E) = \widetilde{E}$. We define*

$$\mathsf{Pred}_b(\widetilde{E}) := \{E : T_b(E) = \widetilde{E}\}, \quad \mathsf{Pred}(\widetilde{E}) := \bigcup_{b \in \{\pm 1\}} \mathsf{Pred}_b(\widetilde{E}).$$

Lemma 3 (Exact predecessors). *Let $\widetilde{E} = E_{\widetilde{A},\widetilde{C}}$ and set $D = \widetilde{A}^2 - 4\widetilde{C}$. If D is not a square in $\mathbb{F}_{p^2}$, then $\mathsf{Pred}(\widetilde{E}) = \varnothing$; otherwise, let $\beta \in \mathbb{F}_{p^2}$ satisfy $\beta^2 = D$. For $\sigma \in \{\pm 1\}$, define*

$$z_\sigma = \frac{\widetilde{A} + \sigma\beta}{8}, \qquad E_\sigma = E_{\frac{\widetilde{A} - 3\sigma\beta}{4}, z_\sigma^2}. \tag{11}$$

Every predecessor of $\widetilde{E}$ is one of E_{+1} and E_{-1}. Moreover, E_σ maps to $\widetilde{E}$ under the label

$$s_\sigma = \frac{z_\sigma}{\mathsf{rt}(z_\sigma^2)} \in \{\pm 1\}. \tag{12}$$

Therefore,

$$\mathsf{Pred}_b(\widetilde{E}) = \{E_\sigma : \sigma \in \{\pm 1\},\ s_\sigma = b\}.$$

Note that β by $-\beta$ only exchanges the two possible predecessors.

Proof. Suppose that $E_{A,C} \in \mathsf{Pred}(\widetilde{E})$ and set $\delta = \widetilde{A} - A$. For the corresponding forward label $b \in \{\pm 1\}$, one has $\delta = 6b \cdot \mathsf{rt}(C)$ and hence $C = \delta^2/36$. Substituting $A = \widetilde{A} - \delta$ into the second coordinate of the radical recurrence yields

$$4\delta^2 - 6\widetilde{A}\delta + 9\widetilde{C} = 0.$$

Its discriminant is $36(\widetilde{A}^2 - 4\widetilde{C}) = 36D$. Thus no predecessor exists if D is not a square. Otherwise, solving for δ and substituting $A = \widetilde{A} - \delta$ and $C = \delta^2/36$ gives the two states in Eq. (11).

For either candidate, the first coordinate of the forward recurrence requires

$$b \cdot \mathsf{rt}(z_\sigma^2) = z_\sigma.$$

Its forward label is therefore exactly the value s_σ in Eq. (12), which proves the stated characterization of $\mathsf{Pred}_b(\widetilde{E})$.

Now we give an example of the existence of the same-sign predecessors.

Example 1 (Same-label predecessors). Work over $\mathbb{F}_{1019^2} = \mathbb{F}_{1019}(\iota)$, where $\iota^2 = -1$, using the canonical square-root rule of Levin and Pedersen. Since 1, 3/4, and 1/4 are quadratic residues in $\mathbb{F}_{1019}$, we have

$$\mathsf{rt}(-1) = \iota, \qquad \mathsf{rt}(-9/16) = \frac{3}{4}\iota, \qquad \mathsf{rt}(-1/16) = \frac{1}{4}\iota.$$

Direct substitution into the radical recurrence gives

$$T_{+1}(E_{\iota,-1}) = T_{+1}\left(E_{\frac{5}{2}\iota,-\frac{9}{16}}\right) = E_{7\iota,-12}; \;\; T_{-1}(E_{\iota,-1}) = T_{-1}\left(E_{-\frac{7}{2}\iota,-\frac{1}{16}}\right) = E_{-5\iota,-4}.$$

By Lemma 3, these are all the predecessors of the respective curves. Therefore,

$$\mathsf{Pred}_{+1}(E_{7\iota,-12}) = \left\{E_{\iota,-1}, E_{\frac{5}{2}\iota,-\frac{9}{16}}\right\}, \qquad \mathsf{Pred}_{-1}(E_{7\iota,-12}) = \varnothing,$$
$$\mathsf{Pred}_{-1}(E_{-5\iota,-4}) = \left\{E_{\iota,-1}, E_{-\frac{7}{2}\iota,-\frac{1}{16}}\right\}, \quad \mathsf{Pred}_{+1}(E_{-5\iota,-4}) = \varnothing.$$

Thus the two curves exhibit predecessor-label patterns $++$ and $--$, respectively.

The Distribution of the Pattern. We discuss the distribution of the patterns of the predecessors here. We provide the experimental results and the provable result under heuristic.

Experimental Results. Based on 10,000 randomly sampled radical-CGL walks, we observed that the patterns $\{+,+\}$ and $\{-,-\}$ each occurred with a frequency of approximately 25%, whereas the mixed-sign pattern $\{+,-\}$ occurred with a frequency of approximately 50%.

Provability under Heuristic. We can explain this distribution under an idealized uniformity assumption. Suppose that $(\widetilde{A}, \widetilde{C})$ is uniformly distributed among valid coefficient pairs for which $D = \widetilde{A}^2 - 4\widetilde{C}$ is a nonzero square in $\mathbb{F}_{p^2}$. Choose either root $\beta^2 = D$ uniformly; changing its sign only exchanges the two predecessors and therefore does not change their unordered label pattern (Lemma 3).

Using the notation of Eq. (11), the inverse relations are

$$\widetilde{A} = 4(z_+ + z_-), \qquad \beta = 4(z_+ - z_-), \qquad \widetilde{C} = 16 z_+ z_-.$$

It follows that (z_+, z_-) is uniformly distributed over the ordered pairs of distinct nonzero elements of $\mathbb{F}_{p^2}$. Indeed, $\widetilde{C} \neq 0$ excludes $z_+ = 0$ and $z_- = 0$, while $D \neq 0$ excludes $z_+ = z_-$.

By Eq. (12), replacing a nonzero z by $-z$ reverses its label. Hence exactly half of the nonzero elements of $\mathbb{F}_{p^2}$ have label $+1$, and the other half have label -1. Among all ordered pairs of distinct nonzero elements, the numbers having patterns $\{++\}$, $\{--\}$, and $\{+-\}$ are, respectively,

$$\frac{(p^2-1)(p^2-3)}{4}, \qquad \frac{(p^2-1)(p^2-3)}{4}, \qquad \frac{(p^2-1)^2}{2}.$$

Since the total number of such ordered pairs is $(p^2-1)(p^2-2)$, we obtain $\Pr[++] = \Pr[--] = \frac{p^2-3}{4(p^2-2)} = \frac{1}{4} + O(p^{-2})$, and $\Pr[+-] = \frac{p^2-1}{2(p^2-2)} = \frac{1}{2} + O(p^{-2})$.

However, we remark that this uniform model does not directly apply to the actual case, since the coefficient pairs produced by radical-CGL walks represent *supersingular curves* and are obtained from a structured recurrence rather than uniform sampling from $\mathbb{F}_{p^2}^2$. Therefore, the distribution is not statistically indistinguishable from the uniform distribution. We therefore make the heuristic assumption that, for curves reached by sufficiently long random walks, the two predecessor labels are unbiased and independent. Under this heuristic assumption, for either fixed $b \in \{\pm 1\}$, $|\mathsf{Pred}_b(\widetilde{E})|$ is approximately distributed as Binomial$(2, 1/2)$ and hence takes the values 0, 1, and 2 with probabilities $1/4 + O(p^{-2}), 1/2 + O(p^{-2}), 1/4 + O(p^{-2})$, respectively.

3.4 Multi-instance backtracking

The inversion procedure of Section 3.3 does not necessarily recover the last secret label from a single evaluation. For a fixed label $b \in \{\pm 1\}$, a state may have two b-labeled predecessors, one lying on the true walk and the other forming an off-path branch. Repeating the inversion on a single evaluation may therefore produce an increasing set of candidate states.

Therefore, the attack will use the evaluation oracle provided in the (residual) pseudorandomness notion Definition 3. That is, the adversary chooses messages $\overrightarrow{m_1}, \ldots, \overrightarrow{m_Q}$ and obtains the evaluation pairs

$$(E_{\overrightarrow{m_j}}, E_{\overrightarrow{m_j}, \overrightarrow{k}}), \qquad E_{\overrightarrow{m_j}} = \mathsf{CGL}(E_0, \overrightarrow{m_j}), \qquad E_{\overrightarrow{m_j}, \overrightarrow{k}} = \mathsf{CGL}(E_{\overrightarrow{m_j}}, \overrightarrow{k}), \tag{13}$$

for $j \in \{1, \ldots, Q\}$. Although the starting curves $E_{\overrightarrow{m_j}}$ differ, all evaluations apply the same secret label sequence $\overrightarrow{k}$. Consequently, every suffix of $\overrightarrow{k}$ must be simultaneously compatible with all Q evaluation curves.

For a candidate suffix $w \in \{\pm 1\}^*$, define the reverse candidate set associated with the j-th evaluation by

$$\mathcal{B}_j(w) := \{E : \mathsf{CGL}(E, w) = E_{\overrightarrow{m_j}, \overrightarrow{k}}\}, \qquad \mathcal{B}_j(\perp) := \{E_{\overrightarrow{m_j}, \overrightarrow{k}}\}. \tag{14}$$

where $\perp$ represents an empty string. If a label $b \in \{\pm 1\}$ is prepended to w, the corresponding candidate set is updated by

$$\mathcal{B}_j(b \parallel w) = \bigcup_{E \in \mathcal{B}_j(w)} \mathsf{Pred}_b(E). \tag{15}$$

A candidate suffix is discarded once $\mathcal{B}_j(b \| w) = \varnothing$ holds for at least one evaluation j. Thus, an incorrect suffix survives only when it admits a possible reverse path for every queried evaluation. We formalize the concept as follows.

Lemma 4 (No false pruning). *Let w be a suffix of the true secret walk $\overrightarrow{k}$. Then $\mathcal{B}_j(w) \neq \varnothing$ for every $j \in \{1, \ldots, Q\}$. Consequently, the multi-instance backtracking procedure never removes a suffix of $\overrightarrow{k}$.*

Proof. Write $\overrightarrow{k} = u \parallel w$. For each evaluation pair in Eq. (13), let

$$E_j^\star := \mathsf{CGL}(E_{\overrightarrow{m_j}}, u).$$

Then

$$\mathsf{CGL}(E_j^\star, w) = \mathsf{CGL}(E_{\overrightarrow{m_j}}, u \parallel w) = E_{\overrightarrow{m_j}, \overrightarrow{k}},$$

and hence $E_j^\star \in \mathcal{B}_j(w)$. Therefore every candidate set associated with a true suffix is nonempty.

Equivalently, during the iterative update in Eq. (15), the state immediately preceding each true label belongs to the corresponding labeled-predecessor set. The real reverse path is therefore retained at every iteration, so the true suffix is never pruned off. □

BacktrackSuffixes$((Y_j)_{j=1}^{Q}, d)$

$\mathcal{L} \leftarrow \{(\perp, (\{Y_j\})_{j=1}^{Q})\}$
for $r = 1, \ldots, d$ **do**
  $\mathcal{L}' \leftarrow \varnothing$
  for $(w, (B_j)_{j=1}^{Q}) \in \mathcal{L}$ **do**
    for $b \in \{\pm 1\}$ **do**
      $B'_j \leftarrow \bigcup_{E \in B_j} \mathsf{Pred}_b(E)$ for each j
      if $B'_j \neq \varnothing$ for every j **then**
        $\mathcal{L}' \leftarrow \mathcal{L}' \cup \{(b \parallel w, (B'_j)_{j=1}^{Q})\}$
      endif
    endfor
  endfor
  $\mathcal{L} \leftarrow \mathcal{L}'$
endfor
return $\mathcal{L}$

Fig. 3. Multi-instance backtracking for candidate suffixes of length d.

The main backtracking procedure in summerzied in Figure 3, the algorithm maintains one reverse candidate set for each evaluation. When a true intermediate curve has two predecessors carrying the correct label, both are retained because the inverse relation alone does not identify which predecessor lies on the original walk. These off-path states may allow an incorrect suffix to survive for one evaluation. Such a suffix, however, must maintain a nonempty candidate set independently for every evaluation. Increasing Q therefore strengthens the pruning condition without risking removal of the true suffix. We give a visualized real exmple in Figure 5.

Optimization. The procedure's memory will grow fast if we store the entires list $\mathcal{L}$. Instead, we can simply store the labels along the path and each leaves.

Also, to reduce the memory burden, backtracking need not cover the entire secret walk. We can choose a parameter c as the checkpoint length. We first recover candidate suffixes of length $n - c$ and then enumerate the remaining c-label prefix in the forward direction. For a candidate decomposition $k = u \parallel v$, where $u \in \{\pm 1\}^c$ and $v \in \{\pm 1\}^{n-c}$, the checkpoint reached from $E_{\overrightarrow{m_j}}$ under u must belong to the reverse candidate set associated with v. Candidates passing this checkpoint test are verified against all evaluation pairs.

$\mathsf{RecoverWalk}(E_{\overrightarrow{k}}, (X_j, Y_j)_{j=1}^{Q}, c)$

$\mathcal{L} \leftarrow \mathsf{BacktrackSuffixes}((Y_j)_{j=1}^{Q}, n-c)$
for $(v, (B_j)_{j=1}^{Q}) \in \mathcal{L}$ **do**
  for $u \in \{\pm 1\}^c$ **do**
    if $\mathsf{CGL}(X_j, u) \in B_j$ for every j **then**
      $\overrightarrow{k}' \leftarrow u \,\|\, v$
      if $\mathsf{CGL}(E_0, \overrightarrow{k}') = E_{\overrightarrow{k}}$ and $\mathsf{CGL}(X_j, \overrightarrow{k}') = Y_j$ for every j **then**
        return $\overrightarrow{k}'$
      endif
    endif
  endfor
endfor
return $\perp$

Fig. 4. Secret-walk recovery by reverse suffix search of length $n-c$, forward checkpoint matching of length c, and final verification against both the public key and all evaluation pairs.

Theorem 2 (Correctness of key recovery). *Let* $E_{\overrightarrow{k}} = \mathsf{CGL}(E_0, \overrightarrow{k})$ *and let* $(X_j, Y_j) = \big(E_{\overrightarrow{m_j}}, E_{\overrightarrow{m_j}, \overrightarrow{k}}\big)$, *for* $j \in \{1, \ldots, Q\}$, *be evaluation pairs generated using the same secret walk* $\overrightarrow{k} \in \{\pm 1\}^n$. *If* $\overrightarrow{k}' \leftarrow \mathsf{RecoverWalk}\big(E_{\overrightarrow{k}}, (X_j, Y_j)_{j=1}^{Q}, c\big)$ *then*

$$\mathsf{CGL}(E_0, \overrightarrow{k}') = E_{\overrightarrow{k}} \qquad \textit{and} \qquad \mathsf{CGL}(X_j, \overrightarrow{k}') = Y_j$$

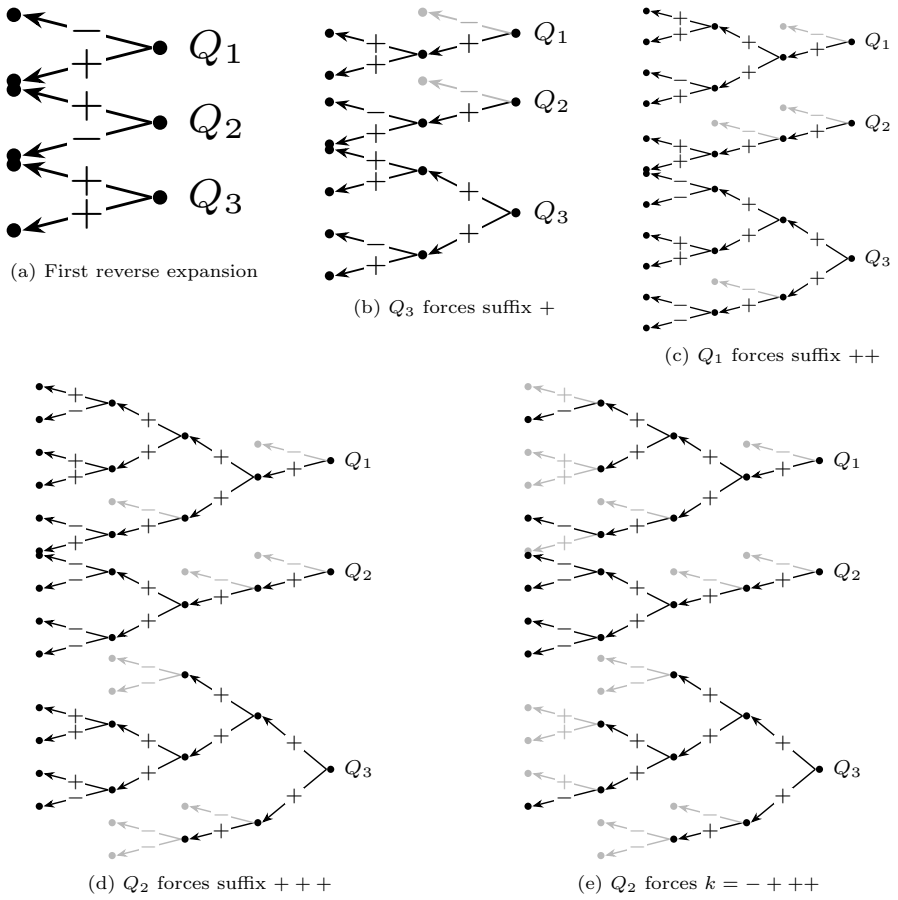
for every $j \in \{1, \ldots, Q\}$. *Moreover, the true walk* $\overrightarrow{k}$ *is among the candidates examined by the algorithm. In particular, if the public key and the supplied evaluation pairs uniquely determine a walk in* $\{\pm 1\}^n$, *then the algorithm returns* $\overrightarrow{k}$.

Proof. Write $\overrightarrow{k} = u^\star \,\|\, v^\star$, $u^\star \in \{\pm 1\}^c$, $v^\star \in \{\pm 1\}^{n-c}$. By Lemma 4, the suffix $v^\star$ survives $\mathsf{BacktrackSuffixes}$. For every j, its reverse candidate set contains the real intermediate curves $\mathsf{CGL}(X_j, u^\star)$, because

$$\mathsf{CGL}\big(\mathsf{CGL}(X_j, u^\star), v^\star\big) = \mathsf{CGL}(X_j, \overrightarrow{k}) = Y_j.$$

Hence the iteration with $(u, v) = (u^\star, v^\star)$ passes the checkpoint test and reaches the final verification.

The algorithm returns a walk $\overrightarrow{k}'$ only after explicitly checking both $\mathsf{CGL}(E_0, \overrightarrow{k}') = E_{\overrightarrow{k}}$ and $\mathsf{CGL}(X_j, \overrightarrow{k}') = Y_j$ for every j. Therefore every returned walk is consistent with both the public key and all supplied evaluation pairs. If these values uniquely determine a walk in $\{\pm 1\}^n$, this consistency implies $\overrightarrow{k}' = \overrightarrow{k}$. □



(a) First reverse expansion

(b) Q_3 forces suffix $+$

(c) Q_1 forces suffix $++$

(d) Q_2 forces suffix $+++$

(e) Q_2 forces $k = -+++$

Fig. 5. An exact four-step backtracking from a real experiment of Figure 4, whose secret walk is $k = (-,+,+,+)$. The rightmost dot Q_i in each row is the output evaluation curve from the query. Black edges remain compatible with the current status, whereas gray edges mean the candidates have been removed. The four pruning rounds are caused by Q_3, Q_1, Q_2, Q_2, respectively.

While it is nature to break the pseudorandomness experiment when the secret key is available, we make a formal summary as follows.

Theorem 3 (Residual-pseudorandomness attack from key recovery). *Suppose that, after Q evaluation queries, a PPT execution of*

$$\mathsf{RecoverWalk}\big(E_{\overrightarrow{k}}, (X_j, Y_j)_{j=1}^{Q}, c\big)$$

returns the secret walk $\overrightarrow{k}$ with probability $\varepsilon(\lambda)$. Then the coefficient-explicit radical-CGL VRF can be distinguished in the residual-pseudorandomness experiment (Definition 3) with advantage at least $\varepsilon(\lambda)\left(\frac{1}{2} - \frac{1}{2^{2\lambda+1}}\right)$.

Proof. The first-stage adversary chooses distinct messages $\overrightarrow{m_1}, \ldots, \overrightarrow{m_Q}$ and queries the evaluation oracle on each of them. From the returned proofs, it obtains the evaluation curves $Y_j = E_{\overrightarrow{m_j}, \overrightarrow{k}}$, and it locally computes

$$X_j = E_{\overrightarrow{m_j}} = \mathsf{CGL}(E_0, \overrightarrow{m_j}).$$

It then runs $\overrightarrow{k}' \leftarrow \mathsf{RecoverWalk}\big(E_{\overrightarrow{k}}, (X_j, Y_j)_{j=1}^{Q}, c\big)$ and selects any fresh challenge message $\overrightarrow{m}^* \notin \{\overrightarrow{m_1}, \ldots, \overrightarrow{m_Q}\}$.

After receiving the challenge value h_b, the second-stage adversary outputs a uniformly random bit if $\overrightarrow{k}' = \perp$. Otherwise, it computes

$$X^* = \mathsf{CGL}(E_0, \overrightarrow{m}^*), \qquad Y^* = \mathsf{CGL}(X^*, \overrightarrow{k}'),$$

queries the random oracle to obtain $h^* = \mathsf{H}(E_{\overrightarrow{k}}, \overrightarrow{m}^*, Y^*)$, and outputs

$$b' = \begin{cases} 0, & h_b = h^*, \\ 1, & h_b \neq h^*. \end{cases}$$

Conditioned on $\overrightarrow{k}' = \overrightarrow{k}$, the value Y^* is the real evaluation curve for the fresh message $\overrightarrow{m}^*$. If $b = 0$, then $h_b = h^*$, so the adversary always outputs $b' = 0$. If $b = 1$, then h_b is uniform in $\{0,1\}^{2\lambda}$ and equals h^* with probability $2^{-2\lambda}$. Thus, conditioned on successful key recovery, the adversary wins with probability $1 - \frac{1}{2^{2\lambda+1}}$. On key-recovery failure, it succeeds with probability $1/2$ by guessing. Its total success probability is therefore at least

$$\frac{1}{2} + \varepsilon(\lambda)\left(\frac{1}{2} - \frac{1}{2^{2\lambda+1}}\right),$$

which proves the claimed distinguishing advantage.

Remark 1. We give the full experiment in Section 5 targeting the propose $n = 256$ for NIST-1 security.

3.5 Other Implications

Partial Key Recovery. Interestingly, sometimes even without queries, we can still recover a few bits from the public key itself. Concretely, under a randomized experiment, given the coefficient (A, C) from $E_(A, C) \leftarrow \mathsf{CGL}(E_0, \overrightarrow{k})$ we have around 50% to recover the last bit of k, and, among them, 12% to recover the last second bit. For the latter case, we remark that this does not mean one can recover the last 2- or 4-isogenies. The recovery of the last two bits is because the predecessors at the same level have the same signs. Hence, one cannot recover the exact backtracking isogeny (i.e. the dual isogeny) but is able to know the last two bits of the secret key.

Other CGL-hash Functions. Our attack is not limited to the radical variant of CGL. The critical cornerstone of our attack is to exploit the status-dependent choice of the path (i.e. the kernel is depending on the representation of current curve). The classic CGL hash [4,5] uses the modular polynomial Φ_2 to determine each step of a non-backtracking walk. Given two consecutive vertices j_{i-1} and j_i, and excluding the exceptional cases in which the modular polynomial has repeated roots, removing the backtracking factor $Y - j_{i-1}$ from $\Phi_2(j_i, Y)$ leaves a quadratic polynomial whose two roots correspond to the two admissible next vertices. Once these roots are ordered according to a deterministic convention, the next message bit selects one of them and therefore determines j_{i+1}. However, the original CGL proposal only states that the two roots should be ordered "according to some convention" and does not prescribe a concrete rule. We describe a doable example as follows.

For example, write $\mathbb{F}_{p^2} = \mathbb{F}_p(\iota)$ and represent each element $a + b\iota$ by $(a, b) \in \{0, \ldots, p-1\}^2$. We may order the roots lexicographically, first by their real parts and then, in case of equality, by their imaginary parts, and let bit 0 select the smaller root and bit 1 the larger root. Under this convention, the resulting j-invariant can occasionally reveal the last message bit: for each of the three possible predecessors j_{n-1} of j_n, the vertex j_n is either the smallest, middle, or largest among the three neighbors of j_{n-1}. If it is the smallest, both possible incoming edges to j_{n-1} assign label 0 to the final step; if it is the largest, both assign label 1; and if it is the middle, the two possible incoming edges assign different labels. Consequently, the last bit is uniquely determined whenever j_n is the smallest neighbor for all three candidate predecessors, or the largest neighbor for all three.

By assuming these three ranks are independent and uniformly distributed, this event occurs with probability $2/27$. Recovering earlier bits requires the same label to remain consistent over all deeper reverse histories and is therefore substantially less likely. This leakage depends on the concrete ordering convention and is not an intrinsic property of the abstract CGL hash.

We remark that the partial secret key recovery above does not reduce the concrete security. In the same-label cases, two candidate predecessor states remain, and therefore the reverse-state branching used by a standard meet-in-the-middle path search will not be reduced. The observation should thus be

viewed as message-bit leakage rather than an improvement over the generic meet-in-the-middle attack.

4 Cryptanalysis of the Group-Action-Based VRF

We analyze the polynomial group-action VRF of Section 2.5. The main approach is to extend the structural attack in [8, Section 7.2].

4.1 Differential subgroup-orbit attack

Lai observed that public evaluations permit coefficient recovery by reducing to GAIP over proper subgroups and reconstructing the coefficients via CRT [10]. For residual pseudorandomness, however, exact coefficient recovery is unnecessary. A single evaluation query suffices to confine the evaluation curve at a fresh message to a polynomial-size orbit.

Theorem 4 (One-query subgroup-orbit attack). *Suppose that the message space is $\mathbb{Z}_N$, that H has 2λ-bit outputs, and that a publicly known factorization $N = rS$ is available for some $1 < r = \mathrm{poly}(\lambda)$. Then one evaluation query breaks residual pseudorandomness with advantage at least $\frac{1}{2} - r/2^{2\lambda+1}$ and solves the one-more unpredictability problem with probability at least $1/r$.*

Proof. The adversary queries an arbitrary $a \in \mathbb{Z}_N$ and obtains $E_a = [f(a)]\,E_0$. It then chooses the fresh message $m^* = a + S \bmod N$. Since $N = rS$ and $r > 1$, we have $S \not\equiv 0 \pmod N$, and hence $m^* \neq a$ in $\mathbb{Z}_N$.

Since $X - Y$ divides $f(X) - f(Y)$ over every commutative ring, there exists $u \in \mathbb{Z}_N$ such that $f(a+S) - f(a) = Su \pmod N$. Moreover, because $Sr = N$, the residue $Su \bmod N$ depends only on $u \bmod r$. Therefore,

$$E_{m^*} = [Su]\,E_a \in \mathcal{L}_a := \{[tS]\,E_a : t \in \mathbb{Z}_r\}\,. \tag{16}$$

The set $\{tS : t \in \mathbb{Z}_r\}$ is a subgroup of $\mathbb{Z}_N$, so $\mathcal{L}_a$ is the orbit of E_a under this subgroup and contains at most r distinct curves.

For the residual-pseudorandomness attack, the adversary queries $\mathsf{H}(pk, m^*, E)$ for every distinct $E \in \mathcal{L}_a$ and outputs $b' = 0$ if and only if the challenge value h_b equals one of the resulting values. If $b = 0$, then Eq. (16) implies that the real output is among the candidates, so the adversary always succeeds. If $b = 1$, then h_b is uniform in $\{0,1\}^{2\lambda}$ and matches one of at most r candidate hash values with probability at most $r/2^{2\lambda}$. Hence the adversary succeeds with probability at least $1 - r/2^{2\lambda+1}$ and has distinguishing advantage at least $\frac{1}{2} - r/2^{2\lambda+1}$.

For the one-more unpredictability attack, the adversary samples $t \xleftarrow{\$} \mathbb{Z}_r$ and outputs $\big(m^*, [tS]\,E_a\big)$. There exists at least one $t^* \in \mathbb{Z}_r$ such that $St^* = f(m^*) - f(a) \pmod N$. For every such t^*, the group-action law gives $[t^*S]\,E_a = E_{m^*}$. Since t is uniform in $\mathbb{Z}_r$, the output is correct with probability at least $1/r$. Because m^* was not queried to the evaluation oracle, this constitutes a valid one-more forgery.

The restriction in Eq. (16) is adapted from the novel structural attack on ζ_d-rGAIP in [8, Section 7.2] for their structural key for some ζ_d such that $\zeta_d^d = 1$. Here, since we do not have to solve for a GAIP instantce, enumerating the candidate orbit already suffices to distinguish the VRF output from random.

For the proposed order $N = 2q$, where q is an odd prime [11, Remark 3], Theorem 4 applies with $r = 2$ and $S = q$. After querying any $a \in \mathbb{Z}_N$, the adversary chooses the fresh message $m^* = a + q \bmod N$, for which $\mathcal{L}_a = \{E_a, [q]\,E_a\}$. Evaluating $\mathsf{H}(pk, m^*, E)$ on these two candidates therefore breaks residual pseudorandomness of VRF in Section 2.5 with distinguishing advantage at least $\frac{1}{2} - 2^{-2\lambda}$ and solves the one-more unpredictability problem in Section 2.5 with probability at least $1/2$.

5 Experimental Results

Since the cryptanalysis of the group-action-based VRF is direct, we focus our experimental evaluation on the radical-CGL-based VRF.

5.1 Implementation and setup

We implemented the attacks in Figures 3 and 4 in C. [3]

To show the independence of the recent endomorphism ring attack [13] of complexity $p^{1/3+o(1)}$, our cryptographic-size implementation uses SQIsign Round-2 Level-3 prime

$$p = 65 \cdot 2^{376} - 1, \qquad \mathbb{F}_{p^2} = \mathbb{F}_p[\iota]/(\iota^2 + 1), \tag{17}$$

where p is a 383-bit prime satisfying $p \equiv 3 \pmod 4$.

The implementation follows the canonical square-root selector of Levin in [11]. Namely, for a square root $\alpha \in \mathbb{F}_{p^2}$, the sign is selected according to the quadratic residuosity of $\Re(\alpha)$ whenever $\Re(\alpha) \neq 0$, and according to the quadratic residuosity of $\Im(\alpha)$ otherwise. The attack accesses this rule only through the public predecessor-label function R.

The implementation first performs a uniformly sampled non-backtracking radical-CGL walk of length

$$4\lceil \log_2 p \rceil = 1532$$

from $(0, 1)$. The resulting curve is fixed as the public starting curve of the scheme.

For each chosen-message evaluation, the implementation independently samples a public message walk of length n from the public starting curve, obtaining a known starting curve X_j. It then applies the same secret walk $k \in \{\pm 1\}^n$ to produce

$$Y_j = \mathsf{CGL}(X_j, k).$$

The reverse search therefore receives only the public evaluation pairs $(X_j, Y_j)_{j=1}^{Q}$.

[3] https://github.com/MaverickOtaku/Radical_VRF_ATK

5.2 Key-recovery benchmarks

We evaluate exact key recovery for secret-walk lengths $n \in \{64, 128, 256\}$ over the field in Eq. (17). For each parameter set, the reverse phase backtracks $n - c$ labels, after which the forward phase enumerates all 2^c prefixes and matches them against the surviving reverse frontiers. The reported running times are wall-clock measurements, and the memory figures are maximum resident-set sizes.

Table 1. Exact multi-instance key recovery over $p = 65 \cdot 2^{376} - 1$. Here n is the secret-walk length, Q is the number of chosen-message evaluations, and c is the number of labels enumerated in the forward phase. The experiment runs on an Intel Core Ultra 9 (2.7GHz).

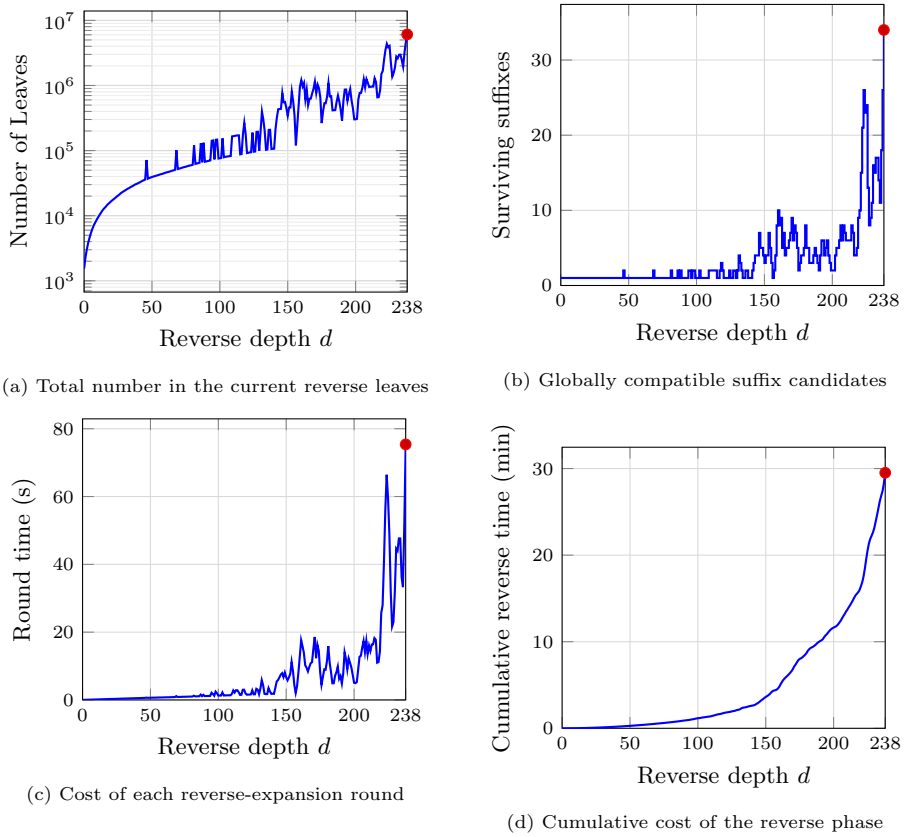
n	Q	c	Time	Peak memory	Recovery
64	64	16	3.6 s	36.1 MiB	unique
128	768	18	1 m 31.2 s	178.5 MiB	unique
256	1536	18	27 m 39.5 s	1.20 GiB	unique

For the parameter $n = 256$, formally proposed in [11], experiment evaluates the attack at the complete secret-walk length of the proposed radical-CGL VRF. It uses $Q = 1536 = 6n$ evaluations, backtracks 238 labels, and enumerates 2^{18} forward prefixes. The reverse phase produces 34 candidate suffixes, of which exactly one extends to a walk consistent with every evaluation pair and the public key. Thus, the full 256-label secret is recovered exactly in under half an hour using approximately 1.20 GiB of memory.

These experiments also illustrate that the attack does not recover each secret label independently from an individual evaluation. Multiple off-path predecessor states may survive for any fixed query. The effective pruning instead comes from requiring one common suffix to remain compatible with all Q evaluation pairs. The resulting number of globally compatible suffixes remains small enough that the final forward enumeration and exact verification isolate the unique secret walk.

6 Conclusion

In this work we analyzed the two isogeny-based VRF constructions proposed in [11]. For the Radical-CGL-based construction, we first showed that storing the public key only as a j-invariant, without additional information fixing its coefficient representation, violates the injectivity premise required by their generic security theorem. We then gave a concrete attack against weak unique provability. Under the coefficient-pair evaluation specified by the Radical-CGL algorithm, we derived algebraic relations between an endpoint curve and its possible predecessors. These relations lead to a full key-recovery attack that recovers the proposed



(a) Total number in the current reverse leaves

(b) Globally compatible suffix candidates

(c) Cost of each reverse-expansion round

(d) Cumulative cost of the reverse phase

Fig. 6. Evolution of the reverse search in the complete $n = 256$ experiment, with $Q = 1536$ evaluation pairs and forward split $c = 18$ to meet in the middle. The horizontal axis is the number d of recovered suffix labels. The implementation stores only the coefficients on the leaves. At the splitting depth $d = n - c = 238$, the reverse search retains 34 globally compatible suffixes represented by 6,081,195 leaves. The final reverse round takes 75.4 seconds, while the complete reverse phase takes less than 30 minutes.

256-bit secret walk using 1536 evaluation queries. The same structure also leaks the last 1 or 2 secret bits from a single coefficient-pair endpoint but not revealing the exact backtracking isogenies. We also show that this is also applicable to the classic CGL with an example. Investigating the applicability to [9] is also an intresting problem.

For the polynomial group-action construction, we extended Lai's observation and showed that one evaluation query confines the output at a related fresh message to a small subgroup orbit. For the proposed group order $N = 2q$, this orbit contains at most two candidates, which breaks residual pseudorandomness with advantage approximately 1/2.

These results show that the security of isogeny-based VRFs depends not only on the hardness of the underlying isogeny problem, but also on the exact curve representation, the algebraic relations exposed by evaluation, and the subgroup structure of the action domain.

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. 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
3. 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
4. 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
5. Chi-Domínguez, J.J., Ochoa-Jiménez, E., Rodas, R.N.P.: Let us walk on the 3-isogeny graph: efficient, fast, and simple. IACR TCHES **2025**(4), 644–666 (2025). https://doi.org/10.46586/tches.v2025.i4.644-666
6. 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
7. Hanzlik, L., Lai, Y.F., Mula, M., Paracucchi, E., Slamanig, D., Tang, G.: Tanuki: New frameworks for (concurrently secure) blind signatures from post-quantum group actions. In: ASIACRYPT 2025, Part IV. pp. 35–69. LNCS, Springer, Singapore (Dec 2025). https://doi.org/10.1007/978-981-95-5113-2_2
8. Katsumata, S., Lai, Y.F., LeGrow, J.T., Qin, L.: CSI-Otter: Isogeny-based (partially) blind signatures from the class group action with a twist. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part III. LNCS, vol. 14083, pp. 729–761. Springer, Cham (Aug 2023). https://doi.org/10.1007/978-3-031-38548-3_24
9. Kunzweiler, S., Maino, L., Moriya, T., Petit, C., Pope, G., Robert, D., Stopar, M., Ti, Y.B.: Radical 2-isogenies and cryptographic hash functions in dimensions 1, 2 and 3. pp. 265–299. LNCS, Springer, Cham (May 2025). https://doi.org/10.1007/978-3-031-91826-1_9
10. Lai, Y.F.: A note on isogeny group action-based pseudorandom functions. Cryptology ePrint Archive, Report 2024/2042 (2024), https://eprint.iacr.org/2024/2042
11. Levin, S., Pedersen, R.: Faster proofs and VRFs from isogenies. In: ASIACRYPT 2025, Part IV. pp. 307–340. LNCS, Springer, Singapore (Dec 2025). https://doi.org/10.1007/978-981-95-5113-2_10
12. 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

13. Wesolowski, B.: The supersingular isogeny problem in time and memory $p^{1/3+o(1)}$. Cryptology ePrint Archive, Paper 2026/1486 (2026), https://eprint.iacr.org/2026/1486

A Potential Fix

As an independent observation and interest, we describe a possible modification to prevent the main attack presented in this work. Since repairing the construction is not our main focus, we do not claim that the resulting scheme satisfies the required security notions.

The main idea is to separate the representation used for the public key from the value exposed by an evaluation. The public key remains the coefficient representation of the terminal curve,

$$pk = (A_{\overrightarrow{k}}, C_{\overrightarrow{k}}),$$

so that it uniquely specifies the output of the coefficient-level Radical-CGL computation. In contrast, if an evaluation produces the curve $E_{A,C}$, the VRF output is computed as

$$y \leftarrow \mathsf{H}\big(pk, m, j(E_{A,C})\big),$$

rather than hashing the coefficient pair (A, C). The accompanying proof must show both that $E_{A,C}$ is the correct evaluation curve and that the hashed value is its j-invariant. Assuming that the proof system is zero-knowledge and reveals no auxiliary information about (A, C), the adversary obtains only $j(E_{A,C})$. The coefficient representation required by our predecessor enumeration is therefore unavailable, and the full key-recovery attack in Section 3.3 no longer applies.

For the curve

$$E_{A,C} : y^2 = x^3 + Ax^2 + Cx,$$

the required value is

$$j(E_{A,C}) = 256\,\frac{(A^2 - 3C)^3}{C^2(A^2 - 4C)}.$$

The proof relation can enforce this computation by checking that the denominator is nonzero and verifying the corresponding cross-multiplied identity. Using the same $\mathbb{F}_{p^2}$-arithmetic representation as the original proof system, this requires ten additional multiplication constraints.

This modification only prevents attacks that rely on coefficient-pair evaluation endpoints. It does not remove leakage already present in the coefficient-valued public key, nor does it constitute a complete security proof for the modified construction.