A New Class of PRFs and Efficient VRFs from Isogeny-based Group Actions *

Junbo Yang[1]✉, Willy Susilo[1], Dung Hoang Duong[1], Partha Sarathi Roy[1], Yannan Li[1], Kazuhide Fukushima[2], and Shinsaku Kiyomoto[2]

[1] Institute of Cryptology and Cybersecurity (iC^2), School of Computing and Information Technology, University of Wollongong, Australia
jy559@uowmail.edu.au
{wsusilo,hduong,partha,yannan}@uow.edu.au
[2] Advanced Technology Laboratories, KDDI Research, Inc., Japan
{ka-fukushima,sh-kiyomoto}@kddi.com

Abstract. Polynomial-in-exponent pseudorandom functions (Poly-style PRFs) based on group actions have recently emerged as a promising framework for post-quantum cryptography, offering compact keys and efficient verifiability. However, the security of these constructions over composite-order class groups remains poorly understood. Prior work suggested that security is governed by the largest prime factor of the group order. In this work, we show that this intuition is flawed. We present two generic attacks demonstrating that the effective security of Poly-style PRFs is determined instead by the smallest prime factor of the group order. As a consequence, existing parameter choices, including CSIDH-512 and SCALLOP-based settings used in recent VRF constructions, are insecure. Our enhanced attack achieves advantage close to one with complexity $O(\sqrt{p})$, where p is the smallest prime factor of the group order. To address this inherent vulnerability, we introduce a new design paradigm, *restrict-then-expand*, and formalize a general class of multivariate polynomial PRFs (MP-PRFs) from group actions. The key idea is to eliminate exploitable structure via carefully restricted input domains, and then recover expressiveness through a provably secure expansion technique. We show that MP-PRFs unify and generalize existing constructions, including Naor-Reingold-style and Poly-style PRFs (with restricted input domains), while achieving security in the generic group action model with twists (GGAM$^\mathsf{T}$) for arbitrary composite-order groups. In addition to this, our construction supports flexible polynomial-size input spaces with fixed output length and admits efficient and compact proofs of correct evaluation. Building on this framework, we obtain new group-action-based VRFs that are both practical and provably secure. At a proof-soundness level close to 128 bits, our VRF instantiation requires 320 group-action evaluations for either proof generation or verification, compared with approximately 69,000–109,500 group-action evaluations in prior work, while retaining short keys and proofs. Overall, our results establish a robust and efficient approach to constructing post-quantum

* This article is the full version of a paper accepted at ASIACRYPT 2026.

PRFs and VRFs from group actions, resolving fundamental limitations of existing Poly-style designs.

Keywords: Cryptographic group actions · Pseudorandom Functions · Verifiable Random Functions · Isogenies

Acknowledgments This work was partially supported by the Australian Research Council (ARC) Linkage Project LP220100332. Willy Susilo was supported by the ARC Australian Laureate Fellowship FL230100033.

Table of Contents

1 Introduction 4
1.1 Our Contributions 5
1.2 Technical Overview 6
2 Preliminaries 9
2.1 Group Actions 10
2.2 Pseudorandom Functions 12
2.3 Verifiable Random Functions 14
3 Poly-style PRFs and Attacks 16
4 The Proposed PRFs 23
4.1 Restricting the Input Space 23
4.2 Sparse-Poly PRFs with Restricted Input Space 28
4.3 The Expansion Technique 33
4.4 (Sparse) Multivariate Polynomial PRFs. 40
5 The Proposed VRFs 41
6 Conclusion and Future Work 53
A Proof of Equivalence of Pseudorandomness Definitions 56
B Additional Details on the Proof of Theorem 4 57
C Formula for QR(n) 64

1 Introduction

Pseudorandom functions (PRFs), first introduced by Goldreich, Goldwasser, and Micali [22], are fundamental cryptographic primitives that underpin a wide range of cryptographic applications, including encryption, digital signatures, authentication, and message integrity. Algebraic PRFs are particularly attractive because their underlying algebraic structures facilitate additional cryptographic functionalities, such as efficient verifiability and distributed evaluation. Several elegant algebraic PRF constructions have been proposed based on hardness assumptions over cyclic groups or the difficulty of integer factorization, such as [34,35,9]. However, with the advent of sufficiently powerful quantum computers, the underlying discrete logarithm and integer factorization problems could be efficiently solved by Shor's algorithm [44]. Consequently, developing efficient algebraic PRFs based on quantum-resistant assumptions is important for post-quantum cryptographic.

Cryptographic group actions have attracted considerable attention in recent years, partly because they provide a unified framework for formulating various computational problems believed to be hard even for quantum adversaries. Examples of cryptographic group actions arise from isogenies [13,16,15], lattice isomorphism [23,18,26], linear code equivalence [43], polynomial isomorphism [38], and equivalence problems involving tensors and trilinear forms [24,46]. Hardness assumptions associated with these group actions have enabled the construction of a wide range of post-quantum cryptographic primitives, including signature schemes [7,8,46], threshold signatures [17], ring and group signatures [6,5], threshold ring signatures [41], blind signatures [27,20], oblivious transfer [30,3], password-authenticated key exchange [1], commitment schemes [14,25], and PRFs and related functional extensions [2,28,42,32,39].

Although cryptographic group actions have demonstrated their power in a variety of cryptographic constructions, the PRF constructions based on group actions remain limited. One notable class of PRFs is Naor-Reingold style PRFs (NR-style PRFs) [2], which, however, have noticeable limitations when considered for practical deployment. In many scenarios, one needs to generate a proof showing that a PRF output y is correctly evaluated from some claimed input x and a committed secret key k; otherwise, according to the definition of pseudorandomness, others cannot efficiently determine whether y is valid. Such PRFs with committed key, which output pseudorandom values along with a proof of correctness, are called Verifiable Random Functions (VRFs). Due to the fact that the number of terms in NR-style PRFs is approximately equal to the input length (typically 128 or 256 bits), the key size, together with the computational cost and proof size of output correctness proofs, grows linearly with the input length. Although techniques such as quadratic twists, unbalanced challenge spaces, and seed trees can partially mitigate these overheads, the resulting efficiency remains unsatisfactory: Capybara [28] requires 109,500 CSIDH-512 group actions [7], amounting to approximately 72 minutes per proof, while Tsubaki [28] requires 69,000 group actions and approximately 45 minutes.

In order to address the aforementioned limitation, a new class of *polynomial-in-exponent PRFs* (hereafter referred to as *Poly-style PRFs*) was introduced

by de Saint Guilhem and Pedersen [42], and has attracted considerable attention. Due to their simpler structure with only a constant number of polynomial terms, they are more suitable for constructing efficient proof systems and supporting potential functionality extensions, including OPRFs [42], VRFs [32], and (threshold) verifiable oblivious pseudorandom functions (VOPRFs) [39]. However, Lai proposed an attack [29] showing that the effective security level of Poly-style PRFs depends merely on the largest prime factor $p_{\max}$ of the composite class group order N. Since constructing large prime-order ideal class groups of quadratic imaginary orders remains challenging, the parameter choices for Poly-style PRFs become highly restricted. A natural idea is therefore to choose parameters such that $p_{\max}$ is sufficiently large. To achieve this goal, Levin and Pedersen [32] subsequently proposed using the SCALLOP approach [16] to construct class groups whose order satisfies $N = f - 1 = 2q$, where both f and q are large primes. Subsequently, Pedersen further explored using the subgroup of order q acting on oriented elliptic curves [39].

In this paper, we present two more direct and efficient attacks, showing that the effective security level of Poly-style PRFs actually depends on the smallest prime factor $p_{\min}$ of N, rather than the largest prime factor $p_{\max}$ suggested by Lai. This implies that our attacks completely break Poly-style PRFs under the CSIDH-512 parameters and under SCALLOP parameters with $N = 2q$. On the other hand, we find that Pederson's prime-order subgroup approach does not fundamentally prevent our attacks and strongly relies on the prime factorization structures of N, and thus lacks generality (see Section 4.2). Meanwhile, recent advances in class group actions on oriented elliptic curves [36,15] are still insufficient for multiple group action operations. These limitations motivate us to seek a more fundamental approach to addressing the above issues:

Under cyclic ideal class group actions, can we construct a new class of secure PRFs applicable to arbitrary composite orders N, while retaining the practicality of Poly-style PRFs?

1.1 Our Contributions

In this paper, we propose two attacks on Poly-style PRFs and systematically study the aforementioned problem. The main contributions of our work are summarized as follows:

1. We propose two efficient attacks[1] against the pseudorandomness of Poly-style PRFs [42] under current practical parameter settings, where the group order N is composite and typically has a small prime factor. Let $p_{\min}$ and $p_{\max}$ denote the smallest and largest prime factors of the group order N, respectively. Our basic

[1] Concurrent work by Lai, Yu, and Zhou [31] presents an attack on Levin and Pedersen's Poly-style-PRF-based VRF that can be viewed as a variant of our attack adapted to their VRF construction. We emphasize that the primary focus of their work is the cryptanalysis of Levin and Pedersen's other, CGL-based VRF construction, whereas our work focuses on developing new techniques for group-action-based PRF constructions. The two works are therefore fundamentally distinct.

attack, using only two oracle queries and one equality test, achieves an advantage close to $1/p_{\min}$, while our enhanced attack, with complexity $O(\sqrt{p_{\min}})$, achieves an advantage close to 1. Under most current practical parameter settings, our attacks are significantly more efficient than Lai's attack [29], which requires solving several GA-Dlog problems of size $p_{\max}$ (e.g., $p_{\min} = 3$ and $\log_2 p_{\max} \approx 135$ for CSIDH-512, while $p_{\min} = 2$ and $\log_2 p_{\max} \approx 256$ for the SCALLOP parameters suggested by Levin and Pedersen [32]).

2. We introduce a novel *restrict-then-expand* technique and construct a new flexible class of MP-PRFs from isogeny-based group actions. They remain secure against both our attacks and Lai's, while simultaneously offering the following features and advantages:

- The class of MP-PRFs is highly general and flexible. Different choices of tuples $(f, \mathcal{S}, l)$ of base polynomials, input spaces, and expansion factors define different subclasses of MP-PRFs. Currently known PRF constructions, including NR-style PRFs [2], Poly-style PRFs [42] (with restricted input spaces), and Pedersen's linear Poly-style PRFs with subgroup input spaces [39], can be viewed as subclasses of our MP-PRFs under particular choices of tuples $(f, \mathcal{S}, l)$.
- Given a cyclic ideal class group action of arbitrary composite order N with known prime factorization, one can tailor the parameters $(f, \mathcal{S}, l)$ using the method provided in Sections 4.2 and 4.4 to obtain Sparse MP-PRFs (SMP-PRFs) that are secure in $\mathrm{GGAM}^{\mathsf{T}}$ under the GA-DDH assumption, resist currently known quantum oracle-query attacks, and simultaneously achieve short keys and efficient, compact proofs of output correctness.
- By adjusting the choices of $\mathcal{S}$ and l, SMP-PRFs can support arbitrary polynomially bounded input lengths. Compared with NR-style PRFs, which also support arbitrary polynomially bounded input lengths, our PRFs enable significantly more efficient verification of output correctness.

3. We propose an efficient VRF scheme based on SMP-PRFs, which is the first VRF construction from group actions that is simultaneously practical, secure, and applicable to arbitrary cyclic ideal class groups. Compared with Capybara and Tsubaki [28], which require 109,500 and 69,000 group action evaluations, respectively, to generate a single proof of correctness, our VRF scheme requires only 320 group action evaluations. Compared with Levin's VRFs [32] based on Poly-style PRFs, which are completely broken by our attacks under most current parameter settings, our scheme no longer relies on the restrictive assumption of prime-order class groups, admits rigorous security proofs, and achieves comparable efficiency. In Table 1, we compare our proposed VRF scheme with existing group-action-based VRFs under CSIDH-512 [7]. Moreover, several optimization techniques can further reduce the proof lengths of our VRFs; their concrete performance is reported in Table 4.

1.2 Technical Overview

Our techniques consist of three main components: (1) the attack on Poly-style PRFs; (2) the construction and security proof of the new SMP-PRFs; and (3)

Table 1. Group-Action-Based VRFs under CSIDH-512 with a 128-Bit Challenge Space

VRFs	$\lvert\mathsf{k}\rvert$	$\lvert\mathsf{vk}\rvert$	$(\lvert x\rvert, \lvert y\rvert)$ (bit)	$\lvert\pi\rvert$	$T_{\mathsf{PG}}/T_{\mathsf{V}}$	Assump./Model
Capybara [28]	2 KB	8.3 KB	(128,512)	39 KB	109500	GA-DDH
Tsubaki [28]	1.3 KB	5.3 KB	(128,512)	34 KB	69000	GA-sDDH
Levin's VRF ($d = 2$) [32]	96 B	160 B	(256,256)	7.5 KB	240	INSECURE
Proposed VRF (128)	96 B	192 B	(128,512)	7.5 KB	320	$\mathrm{GGAM}^{\mathsf{T}}$&GA-DDH
Proposed VRF (256)	128 B	256 B	(256,512)	10 KB	400	$\mathrm{GGAM}^{\mathsf{T}}$&GA-DDH
Proposed VRF (94ℓ)	$32(\ell+1)$ B	$64(\ell+1)$ B	(94ℓ,512)	$2.5(\ell+1)$ KB	$80(\ell+2)$	$\mathrm{GGAM}^{\mathsf{T}}$&GA-DDH

Here, the proposed VRFs are presented in Section 5 and are built upon our SMP-PRFs (16) proposed in Section 4.4. Proposed VRF(n) denotes a scheme with an n-bit input space. $\lvert\mathsf{k}\rvert$, $\lvert\mathsf{vk}\rvert$, $(\lvert x\rvert, \lvert y\rvert)$, and $\lvert\pi\rvert$ denote the lengths of the evaluation key, verification key, input/output pairs, and proof, respectively. T_{PG} and T_{V} denote the number of group actions for a single proof generation or verification.

the VRF scheme based on the SMP-PRFs. For the group action model, Cyclic Effective Group Action with Twists ($\mathrm{CEGA}^{\mathsf{T}}$), used throughout this paper, we refer the reader to Section 2.1.

Efficient attack on Poly-style PRFs. The form of Poly-style PRFs is as follows: given an $\mathrm{CEGA}^{\mathsf{T}}$ $(\mathbb{Z}_N, \mathcal{E}, \diamond, E_0)$, one chooses a fixed positive integer d and randomly selects a polynomial $f(x) \in \mathbb{Z}_N[x]$ of degree at most d, whose coefficient tuple $(a_0, \ldots, a_d)$ serves as the evaluation key $\mathbf{k}$ of the PRF. The evaluation function is then defined as $F(\mathbf{k}, x) = f(x) \diamond E_0 = (a_d x^d + \cdots + a_1 x + a_0) \diamond E_0$, where $x \in \mathbb{Z}_N$ is the input to the PRF. The core idea behind our attack is quite direct: when querying the evaluation oracle on two inputs whose difference is $N/p_{\min}$, an output collision occurs with significantly higher probability for the PRF than for a truly random function. Observe that $f(x_1) - f(x_2) = (a_d(\sum_{i=0}^{d-1} x_1^{d-1-i} x_2^i) + \cdots + a_2(x_1 + x_2) + a_1)(x_1 - x_2)$. Therefore, when we choose two inputs x_1, x_2 such that $x_1 - x_2 = N/p_{\min}$, then $\Pr[f(x_1) \equiv f(x_2) \pmod N \mid \mathbf{k} \xleftarrow{\$} \mathbb{Z}_N^{d+1}] = 1/p_{\min}$. Thus, the gap in collision probability is especially pronounced when N has a small prime factor. At this point, we simply guess that the function is pseudorandom if an output collision occurs, and guess that it is truly random otherwise; this yields an advantage of approximately $1/p_{\min}$ in the pseudorandomness experiment. Moreover, by combining the above approach with the baby-step giant-step technique, we obtain an enhanced attack with complexity $O(\sqrt{p_{\min}})$ and advantage close to 1, which already matches the lower bound on the complexity required for generic group action algorithms to break the pseudorandomness of Poly-style PRFs (see Section 4.1). Therefore, the effective security of Poly-style PRFs actually depends on the smallest prime factor, rather than the largest prime factor claimed by Lai in the note [29].

(Sparse) Multivariate Polynomial PRFs (SMP-PRFs). Our objective is to construct PRFs that are resistant to the above attacks, while remaining efficient and "proof-friendly" (i.e., amenable to the construction of efficient proofs of output correctness). To achieve this goal, we introduce a novel technique, called *Restrict-then-Expand*. As implied by its name, the final PRFs are obtained in two main steps: restricting the valid input space and expanding it.

Restrict the Valid Input Space: The first challenge is how to eliminate the above attacks. Actually, both the proposed attack and Lai's rely on high-gap value queries to the oracle. Therefore, our first step is to restrict the valid input space of Poly-style PRFs to a subset $\mathcal{S}$ of $\mathbb{Z}_N$. Although the ideal case is to restrict the input space such that the difference between any two values in $\mathcal{S}$ is not divisible by any prime factor of N, thereby completely eliminating the above attacks, such a restriction may make the input space too small and affect the efficiency of the VRFs we construct later. Therefore, we allow the existence of two values in this input space whose difference is divisible by some polynomial-size factors of N[2]. In Section 4.1, we rigorously prove the effectiveness of the input-space restriction in $\mathrm{GGAM}^{\mathsf{T}}$ (see Theorem 4). This model is proposed by Duman, Hartmann, and Kiltz [19], which can be regarded as a variant of the Generic Group Model (GGM) [45] in the algebraic setting of group actions.

Using sparse polynomials: To further improve the efficiency and compactness of the PRFs and the corresponding VRFs, we propose using sparse polynomials in place of the original polynomials. We call the resulting PRFs Sparse-Poly PRFs. This optimization relies on the fact that removing some polynomial terms does not affect the security of the PRFs under the $\mathrm{GGAM}^{\mathsf{T}}$, as long as an appropriate valid input space $\mathcal{S}$ is chosen (see Theorem 5). In Section 4.2, we present two candidate sparse polynomials and, taking the CSIDH-512 and SCALLOP-style parameters as examples, select valid input spaces $\mathcal{S}$ for each of them.

Expand the Input Space: Merely restricting the valid input space may result in an excessively small input space (see Table 3), thereby rendering the PRFs impractical. Therefore, our final step in Section 4.3 is to expand the input space. This technique simultaneously endows our final PRFs with a very desirable property — the input space $\mathcal{S}$ can be expanded exponentially to $\mathcal{S}^l$ for any polynomial size positive integer l (called the *expansion factor* in this paper). Therefore, while keeping the output space size fixed, we can freely choose the size of the input space by appropriately selecting $\mathcal{S}$ and l. The technique is as follows: given the base function $F(\mathbf{k}, x) = (c + f_{\mathbf{k}'}(x)) \diamond E_0$ with input space $\mathcal{S} \subseteq \mathbb{Z}_N$, where $\mathbf{k}' \xleftarrow{\$} \mathcal{K}'$, the evaluation key $\mathbf{k} = (c, \mathbf{k}') \xleftarrow{\$} \mathcal{K} = \mathbb{Z}_N \times \mathcal{K}'$, and f is a function from $\mathcal{S}$ to $\mathbb{Z}_N$. We choose an expansion factor l, set the expanded evaluation key space $\mathcal{K}^{\mathsf{ex}} = \mathbb{Z}_N \times \mathcal{K}'^l$. For any key $\mathbf{k}^{\mathsf{ex}} = (c, \mathbf{k}'_1, \ldots, \mathbf{k}'_l) \xleftarrow{\$} \mathcal{K}^{\mathsf{ex}}$, the expanded function is then defined as $F^{\mathsf{ex}}(\mathbf{k}^{\mathsf{ex}}, x) = (c + f_{\mathbf{k}'_1}(x_1) + \cdots + f_{\mathbf{k}'_l}(x_l)) \diamond E_0$, where the input $x = x_1 \| \ldots \| x_l$ with $x_i \in \mathcal{S}$ is from the expanded input space $\mathcal{S}^l$. This expansion technique can be viewed as a structural generalization of NR-style PRFs. Theorem 10 presents a reduction between winning the pseudorandomness game of the function F and that of its expanded function F^{ex} under the GA-DDH assumption. That is, under the GA-DDH assumption, the pseudorandomness of the basic function F implies that of the expanded function F^{ex}. By applying our expansion technique to the Sparse-Poly PRFs, we obtain the final new class of PRFs, which we term *Sparse Multivariate Polynomial PRFs (SMP-PRFs)*.

The proposed VRFs. In Section 5, we propose our efficient VRFs based on

[2] This allowance does not facilitate the attack (including both ours and Lai's).

the SMP-PRFs $F^{\mathsf{SMP}}(\mathbf{k}, x) = (c + a_1 x_1^2 + \cdots + a_l x_l^2) \diamond E_0$, where the evaluation key $\mathbf{k} = (c, a_1, \ldots, a_l)$, the input $x = x_1 \| \cdots \| x_l \in \mathcal{S}^l$ with $x_i \in \mathcal{S}$. This class of PRFs is one of the candidate PRF classes presented in Section 4.3. We choose this class of PRFs because it is sufficiently sparse, *i.e.*, the number of terms in the underlying polynomial is small, which makes our VRFs compact and efficient. A standard approach is to commit to $\mathbf{k}$ and publish the resulting commitment as the verification key. Observing that commitment schemes based on isogeny-based group actions involve two elliptic curves [14], we instead use the pseudorandom evaluations of several fixed public inputs under the key k as the public key. This reduces the public key size by a factor of two. In exchange, the corresponding VRF input space must exclude these constant-many fixed inputs. Specifically, after the evaluation key $\mathbf{k}$ has been sampled uniformly at random, we publish a sequence of pseudorandom values of the SMP-PRFs $F^{\mathsf{SMP}}(\mathbf{k}, \cdot)$ over $l+1$ distinct public input values $u^{(i)}$ $(i = 0, 1, \ldots, l)$ (these input values are part of the public parameters pp), and let the verification key $\mathbf{vk}$ be $(V_0, \ldots, V_l)$, where $V_i = F^{\mathsf{SMP}}(\mathbf{k}, u^{(i)})$. The benefit of publishing pseudorandom values as the verification key lies in the fact that it simplifies the security proof without incurring loss in efficiency and compactness.

The remaining challenge is to construct a zero-knowledge proof for a relation that appears somewhat involved. Specifically, given a CEGA$^{\mathsf{T}}$ $(\mathbb{Z}_N, \mathcal{E}, \diamond, E_0)$, an expansion factor l, $l+1$ input values $\{u^{(i)}\}_{i=0}^{l} = \{u_1^{(i)} \| \cdots \| u_l^{(i)}\}_{i=0}^{l}$, and the corresponding verification key $\mathbf{vk} = (V_0, \ldots, V_l)$, in order to prove the correctness of the output Y on input $x = x_1 \| \ldots \| x_l \in \mathcal{S}^l$, one needs to prove the following relation $\mathcal{R}$:

$$\mathcal{R} = \left\{ \begin{matrix} \mathsf{st} = (\mathsf{pp}, \mathbf{vk}, x, Y); \\ \mathsf{wt} = \mathbf{k} = (c, a_1, \ldots, a_l) \end{matrix} \;\middle|\; \begin{matrix} Y = (c + a_1 x_1^2 + \cdots + a_l x_l^2) \diamond E_0 \quad \wedge \\ V_i = (c + a_1 (u_1^{(i)})^2 + \cdots + a_l (u_l^{(i)})^2) \diamond E_0 \text{ for } i \in [0, l] \end{matrix} \right\}$$

where $\mathsf{pp} = ((\mathbb{Z}_N, \mathcal{E}, \diamond, E_0), l, \{u^{(i)}\}_{i=0}^{l})$ is the public parameter. Our core idea for constructing a proof system for this relation is to transform it to a special case of a subgroup membership proof. A subgroup membership proof over an effective group action $(\mathcal{G}, \mathcal{E}, \star, E_0)$ is a proof system for the following relation:

$$\mathcal{R}_{\hat{\mathcal{G}}} = \left\{ \left(E, E_0, \hat{\mathcal{G}} \right); w \;\middle|\; E = w \star E_0 \wedge w \in \hat{\mathcal{G}} \right\}$$

where $\hat{\mathcal{G}}$ is an exponentially large, efficiently samplable and recognizable subgroup of $\mathcal{G}$. This hard relation can be proven using a concise Sigma protocol. In Section 5, we show that proving the relation $\mathcal{R}$ is equivalent to proving a special instance of the relation $\mathcal{R}_{\hat{\mathcal{G}}}$.

2 Preliminaries

NOTATION. We use $[n]$ to denote the integer set $\{1, 2, \ldots, n\}$, and $[a, b]$ $(a, b \in \mathbb{Z}$ with $a < b)$ to denote the integer set $\{a, a+1, \ldots, b\}$. We denote by $\{0,1\}^k$ the set of all k-bit strings, and by $\{0,1\}^*$ the set of all bit strings of positive integer length. For an integer a and a positive integer N, we use $a \bmod N$ to denote the

set $\{a+kN \mid k \in \mathbb{Z}\}$. We use the boldface lowercase letters such as $\mathbf{a}, \mathbf{b}$ to denote vectors or tuples and the boldface uppercase letters such as $\mathbf{A}, \mathbf{B}$ to denote matrices. Unless otherwise specified, all vectors in this paper are treated as column vectors when involved in matrix operations. For a vector $\mathbf{a}$, $\mathbf{a}[i]$ denotes its i-th component, and $\mathbf{a}[i:j]$ $(i < j)$ denotes the subvector consisting of the components from index i to j. For a finite set $\mathcal{S}$, $|\mathcal{S}|$ denotes its cardinality, and $s \xleftarrow{\$} \mathcal{S}$ denotes that s is sampled uniformly at random from $\mathcal{S}$. For an algorithm $\mathcal{A}$, the notation $y \leftarrow \mathcal{A}(x_1, x_2, \ldots)$ indicates that, on input $x_1, x_2, \ldots$, the algorithm $\mathcal{A}$ outputs y, either deterministically or probabilistically. The notation $\mathcal{A}^{\mathcal{B}}$ denotes an algorithm $\mathcal{A}$ with oracle access to algorithm $\mathcal{B}$. We use $\mathrm{Func}(\mathcal{S}, \mathcal{E})$ to denote the set of all functions from $\mathcal{S}$ to $\mathcal{E}$. For a discrete probability distribution $\mathcal{D}$ over a set $\mathcal{X}$, we define its support as $\mathrm{supp}(\mathcal{D}) := \{x \in \mathcal{X} \mid \Pr_{X \leftarrow \mathcal{D}}[X = x] > 0\}$. Throughout this paper, "PPT" stands for "probabilistic polynomial-time", and "efficient" refers to PPT unless otherwise specified. For two hard assumptions A and B, we write $A \Rightarrow B$ to mean that the hardness of A implies the hardness of B; equivalently, a PPT algorithm for solving B would yield a PPT algorithm for solving A. The Polynomial-in-exponent PRFs introduced by de Saint Guilhem and Pedersen (PKC 2024) are, for brevity, referred to as *Poly-style PRFs*.

2.1 Group Actions

In this section, we briefly introduce several models of cryptographic group actions, which serve as the underlying tool for our construction.

Definition 1 (Group Action, Regular Group Action) *A group $(\mathcal{G}, \cdot)$ is said to act on a set $\mathcal{X}$ if there is a map $\star : \mathcal{G} \times \mathcal{X} \to \mathcal{X}$ that satisfies the following two properties:*
1. *Identity: Let e be the identity element of $\mathcal{G}$, then $e \star X = X$ for any $X \in \mathcal{X}$.*
2. *Compatibility: For any $g, h \in \mathcal{G}$ and $X \in \mathcal{X}$, we have $(g \cdot h) \star X = g \star (h \star X)$*

We adopt the abbreviated notation $(\mathcal{G}, \mathcal{X}, \star)$ to denote a group action. We say that a group action $(\mathcal{G}, \mathcal{X}, \star)$ is regular if, for any $X, Y \in \mathcal{X}$, there exists a unique element $g \in \mathcal{G}$ such that $Y = g \star X$.

To make a mathematically defined group action suitable for cryptographic applications, some additional constraints must be imposed. This leads to the notion of an effective group action (EGA).

Definition 2 (Effective Group Action [2]) *A group action $(\mathcal{G}, \mathcal{X}, \star)$ is said to be effective if the following properties are satisfied:*
1. *$\mathcal{G}$ is finite, and there exist efficient algorithms for membership testing, equality testing, (uniform) random sampling, group operation and inversion.*
2. *The set $\mathcal{X}$ is finite, and there exist efficient algorithms for membership testing and to compute a unique bit-string representation.*
3. *There exists a distinguished element $X_0 \in \mathcal{X}$ which is called origin with known representation.*

4. *There exists an efficient algorithm which, given the bit-string representation of any $g \in \mathcal{G}$ and any $x \in \mathcal{X}$, outputs $g \star x$.*

then we call $(\mathcal{G}, \mathcal{X}, \star, X_0)$ an effective group action (EGA).

Throughout this paper, we use the following group-action model introduced by Duman et al. [19]. Compared with the original definition, we impose the additional assumption that the prime factorization of group order N is publicly known. This assumption is satisfied by the parameter sets currently used to instantiate Poly-style PRFs, including CSIDH-512 and the SCALLOP-based parameter sets adopted by Levin and Pedersen [32,39], for all of which the prime factorization of N is known.

Definition 3 (Cyclic Effective Group Action (with Twists) [19]) *Let $(\mathcal{G}, \mathcal{X}, \star, X_0)$ be an effective group action, then we say it is a cyclic (known-order and known-factorization) effective group action (CEGA) if it satisfies following properties:*

1. *The group $\mathcal{G}$ is cyclic of known order $N \in \mathbb{N}^+$, whose prime factorization is also known.*
2. *There exists a generator $g \in \mathcal{G}$ with known representation (that is, $\mathcal{G} = \langle g \rangle$).*
3. *The group action is regular.*

Additionally, we say that an effective group action $(\mathcal{G}, \mathcal{X}, \star, X_0)$ is a cyclic (known-order) effective group action with twist (CEGA$^\mathsf{T}$) if, in addition to satisfying the above properties, there exists an efficient algorithm that, for any $h \in G$, on input $X = h \star X_0$, outputs $X^\top = h^{-1} \star X_0$.

In a CEGA, the group $(\mathcal{G}, \cdot)$ is isomorphic to $(\mathbb{Z}_N, +)$. Since a generator g of the group is known and fixed, every element $h \in \mathcal{G}$ can be written as $h = g^a$ for a unique $a \in \mathbb{Z}_N$. Throughout this paper, to sample an element $h \in \mathcal{G}$, we sample $a \xleftarrow{\$} \mathbb{Z}_N$ and compute $h \leftarrow g^a$. Therefore, we use the induced action $\diamond : \mathbb{Z}_N \times \mathcal{X} \to \mathcal{X}$ defined by $a \diamond X := g^a \star X$. Correspondingly, in CEGA $(\mathbb{Z}_N, \mathcal{X}, \diamond, X_0)$, we have (1). $0 \diamond X = X$ and (2). $(a+b) \diamond X = a \diamond (b \diamond X)$ for any $X \in \mathcal{X}$. For CEGA$^\mathsf{T}$, we additionally have $(a \diamond X_0)^\top = (-a) \diamond X_0$. Moreover, note that the group action is regular. This allows us to sample an element X uniformly at random from $\mathcal{X}$ by sampling $a \xleftarrow{\$} \mathbb{Z}_N$ and computing $X \leftarrow a \diamond X_0$.

Let $\lambda \in \mathbb{N}^+$ be the security parameter, and let $(\mathbb{Z}_N, \mathcal{X}, \diamond, X_0)$ be a publicly known CEGA$^\top$. We assume that the following problems are hard for every quantum polynomial-time adversary $\mathcal{A}$ given the public description of CEGA$^\top$. For simplicity, we omit the dependence on λ throughout.

Problem 1 (GA-DLog). Sample $a \xleftarrow{\$} \mathbb{Z}_N$ and set $X_1 = a \diamond X_0$. Given (X_0, X_1), recover a.

Problem 2 (GA-CDH). Sample $a, b \xleftarrow{\$} \mathbb{Z}_N$ and set $X_1 = a \diamond X_0$, $Y_0 = b \diamond X_0$. Given (X_0, X_1, Y_0), compute $Y_1 = a \diamond Y_0 = (a+b) \diamond X_0$.

Problem 3 (GA-DDH). Sample $a, b, r \xleftarrow{\$} \mathbb{Z}_N$ and $\delta \xleftarrow{\$} \{0,1\}$, and set $X_1 = a \diamond X_0,\ X_2 = b \diamond X_0,\ Z_0 = r \diamond X_0,\ Z_1 = (a+b) \diamond X_0$. Given $(X_0, X_1, X_2, Z_\delta)$, decide δ.

CSIDH Group Actions. A concrete instantiation of CEGA and CEGA$^{\mathsf{T}}$ can be obtained from the CSIDH group action [13], an isogeny-based group action given by the free and transitive action of the ideal class group $\mathrm{Cl}(\mathfrak{O})$ on the set of $\mathbb{F}_p$-isomorphism classes of supersingular elliptic curves $E/\mathbb{F}_p$ satisfying $\mathrm{End}_{\mathbb{F}_p}(E) \cong \mathfrak{O}$. Here, $\mathfrak{O} = \mathbb{Z}[\pi]$ is an imaginary quadratic order, where π denotes the $\mathbb{F}_p$-Frobenius endomorphism. CSI-FiSh [7] computes the structure of the class group $\mathrm{Cl}(\mathfrak{O})$ associated with the CSIDH-512 parameter set. This class group is cyclic, generated by the ideal class $g = [\mathfrak{l}_1]$, where $\mathfrak{l}_1 = \langle 3, \pi - 1 \rangle$, and has order

$$|\mathrm{Cl}(\mathfrak{O})| = N = 3 \times 37 \times 1407181 \times 51593604295295867744293584889 \\ \times 315994145046819958530082787455878322049 09.$$

CSI-FiSh also computes a suitably reduced basis of the relation lattice, thereby enabling efficient online evaluation of the action of any class-group element. Moreover, the quadratic-twist operation used in CSI-FiSh is efficiently computable: if $E = a \diamond E_0$, then its quadratic twist $E^{\top}$ is $\mathbb{F}_p$-isomorphic to $(-a) \diamond E_0$.

However, subsequent quantum-security analyses suggest that CSIDH-512 may fall short of NIST security level 1, motivating the use of larger parameters [4,11,40]. To address the difficulty of computing the class-group structure for larger parameters, De Feo et al. [16] proposed SCALLOP, an isogeny-based group action in which the class group $\mathrm{Cl}(\mathfrak{O})$ acts freely and transitively on a suitable set of $\mathfrak{O}$-oriented supersingular elliptic curves. Here, $\mathfrak{O} = \mathbb{Z} + f\mathfrak{O}_0$ is a quadratic order of large prime conductor f, where $\mathfrak{O}_0$ is a maximal quadratic order of small discriminant. This choice enables efficient computation of the order and structure of $\mathrm{Cl}(\mathfrak{O})$. Nevertheless, evaluating the group action in the original SCALLOP construction remains computationally expensive. Subsequent works, including KLaPoTi [37] and PEGASIS [15], have developed efficient algorithms for evaluating isogeny-based group actions. For larger-parameter instantiations of known-order CEGAs, Pedersen [39] considered SCALLOP-based cyclic ideal class groups with efficiently computable structure and employed KLaPoTi to evaluate the corresponding group action efficiently. As the technical details of these group action algorithms are beyond the scope of this work, we refer interested readers to the respective papers.

2.2 Pseudorandom Functions

Pseudorandom functions (PRFs) [22] are efficiently computable functions such that no efficient algorithm can distinguish them from a truly random function. The formal definition is as follows.

Definition 4 (Pseudorandom Functions) *Let $\lambda \in \mathbb{N}^+$ be the security parameter, and let $F_\lambda : \mathcal{K}_\lambda \times \mathcal{X}_\lambda \to \mathcal{Y}_\lambda$ be an efficiently computable keyed function, where $\mathcal{K}_\lambda$, $\mathcal{X}_\lambda$, and $\mathcal{Y}_\lambda$ denote the key, input, and output spaces, respectively, and*

the bit lengths of their elements are polynomially bounded in λ. Define the set of functions $\mathcal{F}_\lambda = \{F_\lambda(k,\cdot) : \mathcal{X}_\lambda \to \mathcal{Y}_\lambda \mid k \in \mathcal{K}_\lambda\}$, and let $\mathsf{R} \xleftarrow{\$} \mathrm{Func}(\mathcal{X}_\lambda, \mathcal{Y}_\lambda)$ be a truly random function. We define the oracle $\mathcal{O}^{\mathsf{PR}}_{b,k,\lambda}(\cdot)$ for $b \in \{0,1\}$ as follows.

Oracle $\mathcal{O}^{\mathsf{PR}}_{0,k,\lambda}(\cdot)$	**Oracle $\mathcal{O}^{\mathsf{PR}}_{1,k,\lambda}(\cdot)$**
Input $x \in \mathcal{X}_\lambda$	Input $x \in \mathcal{X}_\lambda$
$y \leftarrow \mathsf{R}(x)$; Return y	$y \leftarrow F_\lambda(k,x)$; Return y

Then, we say that a family of sets of functions $\mathcal{F} := \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}^+}$ is pseudorandom if, for all sufficiently large $\lambda \in \mathbb{N}^+$, and every PPT adversary $\mathcal{A}$, its advantage

$$\mathbf{Adv}^{\mathsf{PR}}_{\mathcal{A},\mathcal{F}}(\lambda) := 2\left(\Pr\left[b^* = b \;\middle|\; \begin{array}{c} b \xleftarrow{\$} \{0,1\},\ k \xleftarrow{\$} \mathcal{K}_\lambda,\ \mathsf{R} \xleftarrow{\$} \mathrm{Func}(\mathcal{X}_\lambda, \mathcal{Y}_\lambda) \\ b^* \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{PR}}_{b,k,\lambda}}(1^\lambda) \end{array}\right] - \frac{1}{2}\right)$$

is a negligible function in λ. Here, the probability is taken over the random choices of b, k, and the truly random function R, as well as the internal randomness of $\mathcal{A}$.

The above definition of pseudorandomness will be used to prove the security of the PRFs obtained via our expansion technique (Section 4.3). In some cases, however, it is more convenient to use other equivalent definitions. Next, we present two equivalent definitions of pseudorandomness. The first will be used in Section 4.1 and 4.2 to analyze the security of our PRFs in the GGAM$^{\mathsf{T}}$ model, while the second will be used in the security proof of our VRFs in Section 5.

Equivalent Definition 1. This equivalent definition has the same structure as Definition 4, with the difference being that the oracles $\mathcal{O}^{\mathsf{PR}}_{b,k,\lambda}(\cdot)$ ($b \in \{0,1\}$) are replaced by the following oracles $\mathcal{O}^{\mathsf{PR}'}_{b,k,\lambda}(\cdot)$. Correspondingly, the adversary $\mathcal{A}$ is now given access to oracle $\mathcal{O}^{\mathsf{PR}'}_{b,k,\lambda}(\cdot)$.

Oracle $\mathcal{O}^{\mathsf{PR}'}_{b,k,\lambda}(\cdot)$
Input $x \in \mathcal{X}_\lambda$; $y_0 \leftarrow \mathsf{R}(x)$, $y_1 \leftarrow F_\lambda(k,x)$; Return (y_b, y_{1-b})

Namely, the oracle returns both the PRF output and the truly random function output, with the bit $b \in \{0,1\}$ determining their order. The equivalence between this variant and Definition 4 is intuitive, and a simple proof can be found in Appendix A.

Equivalent Definition 2. The pseudorandomness defined here is referred to as "*cannot be polynomially inferred*"[3] in [22], and was proven to be equivalent to the

[3] In fact, there is a subtle difference between the notion of "cannot be polynomially inferred" and our definition. Specifically, the former does not allow the adversary to make a second round of oracle queries after outputting the challenge input $\hat{x}$, whereas our definition does. Nevertheless, these two notions are equivalent. For our definition, equivalence to the standard definition (i.e., Definition 4) can be established using a similar proof based on a hybrid argument.

pseudorandomness in Definition 4 (see Theorem 4 in [22]). We adopt the notation from Definition 4, and define a PRF evaluation oracle $\mathcal{O}_{k,\lambda}^{\mathsf{PR}''}(\cdot)$ as follows. The oracle maintains an internal query set $\mathcal{L}_{\mathsf{q}}$ that records all previously queried inputs and is initialized as $\mathcal{L}_{\mathsf{q}} \leftarrow \varnothing$. On input $x \in \mathcal{X}_\lambda$, the oracle updates the set $\mathcal{L}_{\mathsf{q}} \leftarrow \mathcal{L}_{\mathsf{q}} \cup \{x\}$, and then outputs $\mathcal{F}_\lambda(k, x)$. We say that the function family $\mathcal{F} := \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}^+}$ is pseudorandom if, for every PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and all sufficiently large $\lambda \in \mathbb{N}^+$, its advantage $\mathbf{AdvPRF}_{\mathcal{A},\mathcal{F}}^{\mathsf{PR}''}(\lambda)$ with respect to the following experiment $\mathbf{ExpPRF}_{\mathcal{A},\mathcal{F}}^{\mathsf{PR}''}$ (Fig. 1) is negligible in λ, where

$$\mathbf{AdvPRF}_{\mathcal{A},\mathcal{F}}^{\mathsf{PR}''}(\lambda) := 2\left(\Pr[1 \leftarrow \mathbf{ExpPRF}_{\mathcal{A},\mathcal{F}}^{\mathsf{PR}''}(\lambda)] - \frac{1}{2}\right),$$

and the probability is taken over the randomness in the experiment and randomness of $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.

Fig. 1. Experiments for PRFs

$\mathcal{L}_{\mathsf{q}} \leftarrow \varnothing$; $k \xleftarrow{\$} \mathcal{K}_\lambda$; $(\hat{x}, state) \leftarrow \mathcal{A}_1^{\mathcal{O}_{k,\lambda}^{\mathsf{PR}''}}(1^\lambda)$
$\hat{y}_0 \xleftarrow{\$} \mathcal{Y}_\lambda$; $\hat{y}_1 \leftarrow F_\lambda(k, \hat{x})$.
$b \xleftarrow{\$} \{0,1\}$; $b^* \leftarrow \mathcal{A}_2^{\mathcal{O}_{k,\lambda}^{\mathsf{PR}''}}(state, \hat{y}_b)$
If $\hat{x} \in \mathcal{L}_{\mathsf{q}}$, then abort and return $\perp$
Else return $b^* == b$ $\quad$ $\mathbf{ExpPRF}_{\mathcal{A},\mathcal{F}}^{\mathsf{PR}''}(\lambda)$

Remark: The operator "==" denotes an equality test. It outputs 1 *if the two operands are equal, and outputs* 0 *otherwise.*

2.3 Verifiable Random Functions

Verifiable random functions (VRFs) extend pseudorandom functions by associating each output with a publicly verifiable proof of correct evaluation, while ensuring that outputs on unqueried inputs remain pseudorandom. In this paper, we adopt the standard definition in [33,28].

Definition 5 (Verifiable Random Functions) *Let λ be the security parameter, and let $\mathcal{X}_\lambda$, $\mathcal{Y}_\lambda$, $\mathcal{K}_\lambda$, and $\mathcal{V}_\lambda$ denote the input, output, secret-key, and verification-key spaces, respectively, where the bit lengths of their elements are polynomially bounded in λ. A verifiable random function (VRF) scheme Π_{VRF} consists of four PPT algorithms* $(\mathsf{VRF.Setup}, \mathsf{VRF.KeyGen}, \mathsf{VRF.Eval}, \mathsf{VRF.Verify})$ *such that*

- $\mathsf{VRF.Setup}(1^\lambda) \to \mathsf{pp}$: *On input a security parameter 1^λ, this probabilistic algorithm outputs some global, public parameter* pp.
- $\mathsf{VRF.KeyGen}(\mathsf{pp}) \to (\mathsf{sk}, \mathsf{vk})$: *On input the public parameter* pp*, this probabilistic algorithm outputs two binary strings, a secret key $\mathsf{sk} \in \mathcal{K}_\lambda$ and a verification key $\mathsf{vk} \in \mathcal{V}_\lambda$.*

- $\mathsf{VRF.Eval}(\mathsf{pp}, \mathsf{sk}, \mathsf{vk}, x) \to (y, \pi)$: *On input the public parameter* pp*, a secret key* $\mathsf{sk} \in \mathcal{K}_\lambda$*, a verification key* $\mathsf{vk} \in \mathcal{V}_\lambda$*, and an input* $x \in \mathcal{X}_\lambda$*, this algorithm outputs a VRF value* $y \in \mathcal{Y}_\lambda$ *and a corresponding proof* π *proving the correctness of* y*.*
- $\mathsf{VRF.Verify}(\mathsf{pp}, \mathsf{vk}, x, y, \pi) \to \{0, 1\}$: *On input public parameters* pp*, a verification key* $\mathsf{vk} \in \mathcal{V}_\lambda$*, an input* $x \in \mathcal{X}_\lambda$*, a VRF value* $y \in \mathcal{Y}_\lambda$*, and a corresponding proof* π*, this algorithm outputs either* 1 *(accept) or* 0 *(reject).*

Additionally, a VRF scheme $\Pi_{\mathsf{VRF}} = (\mathsf{VRF.Setup}, \mathsf{VRF.KeyGen}, \mathsf{VRF.Eval}, \mathsf{VRF.Verify})$ should satisfy complete provability, unique provability, and residual pseudorandomness. The definitions of these properties are given below.

Complete Provability. A VRF scheme Π_{VRF} is said to have complete provability if for every $\lambda \in \mathbb{N}^+$ and every $x \in \mathcal{X}_\lambda$,

$$\Pr\left[\mathsf{VRF.Verify}(\mathsf{pp}, \mathsf{vk}, x, y, \pi) = 1 \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{VRF.Setup}(1^\lambda), \\ (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{VRF.KeyGen}(\mathsf{pp}), \\ (y, \pi) \leftarrow \mathsf{VRF.Eval}(\mathsf{pp}, \mathsf{sk}, \mathsf{vk}, x) \end{array}\right] = 1.$$

Unique Provability. A VRF scheme Π_{VRF} is said to satisfy unique provability if, for every PPT[4] adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the advantage of $\mathcal{A}$ in the following experiment $\mathbf{ExpVRF}^{\mathsf{UP}}_{\mathcal{A},\Pi_{\mathsf{VRF}}}$ (Fig. 2) is negligible in λ, where the advantage of $\mathcal{A}$ is defined as

$$\mathbf{AdvVRF}^{\mathsf{UP}}_{\mathcal{A},\Pi_{\mathsf{VRF}}}(\lambda) := \Pr[\mathbf{ExpVRF}^{\mathsf{UP}}_{\mathcal{A},\Pi_{\mathsf{VRF}}}(1^\lambda) = 1],$$

and the probability is taken over the randomness used by $\mathcal{A}$ and the randomness used in the experiment.

Fig. 2. Experiments for Unique Provability and (Residual) Pseudorandomness

$\mathsf{pp} \leftarrow \mathsf{VRF.Setup}(1^\lambda)$
$(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathcal{A}_1(\mathsf{pp})$
$(x, y_1, \pi_1, y_2, \pi_2) \leftarrow \mathcal{A}_2(\mathsf{pp}, \mathsf{sk}, \mathsf{vk})$
If $\mathsf{VRF.Verify}(\mathsf{pp}, \mathsf{vk}, x, y_1, \pi_1) = 1$
$\wedge\ \mathsf{VRF.Verify}(\mathsf{pp}, \mathsf{vk}, x, y_2, \pi_2) = 1$
$\wedge\ y_1 \neq y_2$ then return 1
Else return 0

$\boxed{\mathbf{ExpVRF}^{\mathsf{UP}}_{\mathcal{A},\Pi_{\mathsf{VRF}}}(\lambda)}$

$\mathcal{L}_{\mathsf{q}} \leftarrow \varnothing$; $\mathsf{pp} \leftarrow \mathsf{VRF.Setup}(1^\lambda)$
$(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{VRF.KeyGen}(\mathsf{pp})$
$(\hat{x}, state) \leftarrow \mathcal{A}_1^{\mathcal{O}^{\mathsf{VRF}}_{\mathsf{sk},\lambda}}(\mathsf{pp}, \mathsf{vk})$
$y_0 \xleftarrow{\$} \mathcal{Y}_\lambda$; $(y_1, \pi) \leftarrow \mathsf{VRF.Eval}(\mathsf{pp}, \mathsf{sk}, \mathsf{vk}, \hat{x})$
$b \xleftarrow{\$} \{0, 1\}$; $b^* \leftarrow \mathcal{A}_2^{\mathcal{O}^{\mathsf{VRF}}_{\mathsf{sk},\lambda}}(y_b, state)$
If $\hat{x} \in \mathcal{L}_{\mathsf{q}}$, then abort and return $\perp$
Else return $b^* == b$

$\boxed{\mathbf{ExpVRF}^{\mathsf{PR}}_{\mathcal{A},\Pi_{\mathsf{VRF}}}(\lambda)}$

(Residual) Pseudorandomness. Given a VRF scheme Π_{VRF}, define a VRF evaluation oracle $\mathcal{O}^{\mathsf{VRF}}_{\mathsf{sk},\lambda}(\cdot)$ as follows. The oracle maintains an internal query set $\mathcal{L}_{\mathsf{q}}$ that records all previously queried inputs and is initialized as $\mathcal{L}_{\mathsf{q}} \leftarrow \varnothing$. On

[4] For our construction, the adversary in the unique provability experiment can be taken to be computationally unbounded, provided that it makes at most polynomially many queries to the random oracle.

input $x \in \mathcal{X}_\lambda$, it updates the set $\mathcal{L}_{\mathsf{q}} \leftarrow \mathcal{L}_{\mathsf{q}} \cup \{x\}$, and then outputs $(y, \pi) \leftarrow \mathsf{VRF.Eval}(\mathsf{pp}, \mathsf{sk}, \mathsf{vk}, x)$. We say that Π_{VRF} satisfies residual pseudorandomness if for every PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the advantage of $\mathcal{A}$ in the above experiment $\mathbf{ExpVRF}^{\mathsf{PR}}_{\mathcal{A},\Pi_{\mathsf{VRF}}}$ (Fig. 2) is negligible in λ, where the advantage of $\mathcal{A}$ is defined as

$$\mathbf{AdvVRF}^{\mathsf{PR}}_{\mathcal{A},\Pi_{\mathsf{VRF}}}(\lambda) := 2\left(\Pr[1 \leftarrow \mathbf{ExpVRF}^{\mathsf{PR}}_{\mathcal{A},\Pi_{\mathsf{VRF}}}(\lambda)] - \frac{1}{2}\right),$$

and the probability is taken over the randomness used by $\mathcal{A}$ and the randomness used in the experiment.

In Sections 3–5, to simplify notation, we omit the security parameter λ from the subscripts of PRF(VRF)-related notation whenever no ambiguity arises. Both the construction and the security analysis are understood to be implicitly parameterized by a security parameter λ.

3 Poly-style PRFs and Attacks

In this section, we briefly review the Poly-style PRFs introduced by de Saint Guilhem and Pedersen [42] and present our attacks against them. We also compare our algorithm with Lai's approach (see [29]), thereby explaining why our attack represents a significantly more severe threat to Poly-style PRFs.

Poly-style PRFs. Let $(\mathbb{Z}_N, \mathcal{E}, \diamond, E_0)$ be a CEGA$^{\mathsf{T}}$. A degree-d [5] ($d \geqslant 1$) Poly-style PRF $F^{\mathsf{poly}} : \mathcal{K} \times \mathbb{Z}_N \to \mathcal{E}$ is defined by

$$F^{\mathsf{poly}}(\mathbf{k}, x) = f(x) \diamond E_0 = \left(a_d x^d + \cdots + a_1 x + a_0\right) \diamond E_0, \tag{1}$$

where $f(x) = a_d x^d + \cdots + a_1 x + a_0$ is a polynomial over $\mathbb{Z}_N[x]$ and $x \in \mathbb{Z}_N$ is the PRF input. The coefficient vector $\mathbf{k} = (a_0, \ldots, a_d) \in \mathbb{Z}_N^{d+1} =: \mathcal{K}$ serves as the secret evaluation key.

In this section, our basic attack is analyzed under the case that $\mathbf{k} \xleftarrow{\$} \mathcal{K}$, i.e., the coefficient vector is sampled uniformly at random from $\mathbb{Z}_N^{d+1}$. For our enhanced attack, the evaluation key $\mathbf{k}$ may instead be sampled from an arbitrary fixed distribution over the key space $\mathbb{Z}_N^{d+1}$.

The Basic Attack. Given a Poly-style PRF F^{poly} with polynomial degree d and a random secret evaluation key $\mathbf{k} = (a_0, \ldots, a_d) \xleftarrow{\$} \mathbb{Z}_N^{d+1}$, the basic version of our attack $\mathcal{A}_{\mathsf{basic}}$ proceeds as follows.

1. Let $p_{\min}$ be the smallest prime factor of N. $\mathcal{A}_{\mathsf{basic}}$ chooses $i, j \in [0, p_{\min} - 1]$ with $i \neq j$ and $r \in [0, \frac{N}{p_{\min}} - 1]$ arbitrarily, then sets $x_1 = i\frac{N}{p_{\min}} + r$, $x_2 = j\frac{N}{p_{\min}} + r$.

[5] In this paper, "degree-d" does not, strictly speaking, mean that $f(x)$ is necessarily a polynomial of degree exactly d. Rather, it means that the key space is defined as $\mathcal{K} = \mathbb{Z}_N^{d+1}$, and the corresponding key $\mathbf{k} = (a_0, \ldots, a_d)$ is sampled according to some distribution $\mathcal{D}_{\mathcal{K}}$ over $\mathcal{K}$.

2. $\mathcal{A}_{\mathsf{basic}}$ queries the evaluation oracle $\mathcal{O}^{\mathsf{PR}}_{b,\mathbf{k}}$ for F^{poly} on inputs x_1 and x_2, and obtains $Y_1 = \mathcal{O}^{\mathsf{PR}}_{b,\mathbf{k}}(x_1)$, $Y_2 = \mathcal{O}^{\mathsf{PR}}_{b,\mathbf{k}}(x_2)$.
3. If $Y_1 = Y_2$, then $\mathcal{A}_{\mathsf{basic}}$ outputs the guess $b^* = 1$; otherwise, it outputs $b^* = 0$.

Before analyzing the advantage of the basic attack, we introduce the following Lemma 1 that characterizes the number of zeros of an affine multivariate polynomial over $\mathbb{Z}_N$. It can be viewed as a generalization of Lemma 2 in [19] and will serve as an important tool in Sections 3 and 4.

Lemma 1 *For integers $n \geqslant 1$ and $N \geqslant 2$, let $a_1, \ldots, a_n, c \in \mathbb{Z}$. For the congruence $a_1x_1 + \cdots + a_nx_n \equiv c \pmod{N}$, the number of solutions $(x_1, \ldots, x_n) \in \mathbb{Z}_N^n$ is $N^{n-1}\gcd(a_1, \ldots, a_n, N)$ if $\gcd(a_1, \ldots, a_n, N) \mid c$, and 0 otherwise.*

Proof. First, it is straightforward to see that the congruence

$$a_1x_1 + \cdots + a_nx_n \equiv c \pmod{N}$$

has a solution if and only if $\gcd(a_1, \ldots, a_n, N) \mid c$. Suppose that there exists a solution $(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ such that $a_1\widetilde{x}_1 + \cdots + a_n\widetilde{x}_n \equiv c \pmod{N}$. Then necessarily $a_1\widetilde{x}_1 + \cdots + a_n\widetilde{x}_n \equiv c \pmod{\gcd(a_1, \ldots, a_n, N)}$. Note that $a_i \equiv 0 \pmod{\gcd(a_1, \ldots, a_n, N)}$ for $i = 1, \ldots, n$, which implies that $\gcd(a_1, \ldots, a_n, N) \mid c$. Conversely, if $\gcd(a_1, \ldots, a_n, N) \mid c$, there must exist $(\widetilde{x}_1, \ldots, \widetilde{x}_n, \tilde{t}) \in \mathbb{Z}^{n+1}$ such that $\widetilde{x_1}a_1 + \cdots + \widetilde{x_n}a_n + \tilde{t}N = c$, namely, $a_1\widetilde{x}_1 + \cdots + a_n\widetilde{x}_n \equiv c \pmod{N}$.

Next, we start to prove that, if $\gcd(a_1, \ldots, a_n, N) \mid c$, then the number of solutions is $N^{n-1}\gcd(a_1, \ldots, a_n, N)$. We begin with the case $n = 1$, and then complete the proof by mathematical induction on n[6].

– When $n = 1$, the statement is a standard mathematical fact, and the proof is straightforward (see, e.g., Lemma 2 in [19]).
– Assume that the statement holds for $n \leqslant k$. We prove that it also holds for $n = k + 1$. Consider the congruence

$$a_1x_1 + \cdots + a_kx_k \equiv c - a_{k+1}x_{k+1} \pmod{N}. \tag{2}$$

We count the number of distinct values of $x_{k+1} \pmod{N}$ for which the above congruence has a solution in $(x_1, \ldots, x_k)$. Equivalently, this requires

$$a_{k+1}x_{k+1} \equiv c \pmod{\gcd(a_1, \ldots, a_k, N)}.$$

By the statement for $n = 1$, the congruence admits exactly

$$\gcd(a_{k+1}, \gcd(a_1, \ldots, a_k, N)) = \gcd(a_1, \ldots, a_k, a_{k+1}, N)$$

[6] This lemma can also be proved by constructing the group homomorphism $\varphi : \mathbb{Z}_N^n \longrightarrow \mathbb{Z}_N$, $(x_1, \ldots, x_n) \longrightarrow a_1x_1 + \cdots + a_nx_n$. We can observe that the solution set of the congruence is precisely the coset $\widetilde{\mathbf{x}} + \ker\varphi$. Note that $|\widetilde{\mathbf{x}} + \ker\varphi| = |\ker\varphi|$ and $|\operatorname{im}\varphi| = N/\gcd(a_1, \ldots, a_n, N)$. It then follows from the first isomorphism theorem that $|\ker\varphi| = |\mathbb{Z}_N^n|/|\operatorname{im}\varphi| = N^{n-1}\gcd(a_1, \ldots, a_n, N)$.

distinct solutions for $x_{k+1} \pmod{\gcd(a_1, \ldots, a_k, N)}$. Consequently, there are

$$\frac{N}{\gcd(a_1, \ldots, a_k, N)} \cdot \gcd(a_1, \ldots, a_{k+1}, N)$$

distinct solutions for $x_{k+1} \pmod{N}$. Moreover, by the statement for $n = k$, for each such value of $x_{k+1} \pmod{N}$, the congruence (2) admits exactly

$$N^{k-1} \gcd(a_1, \ldots, a_k, N)$$

solutions for $(x_1, \ldots, x_k) \pmod{N}$. Therefore, the total number of solutions $(x_1, \ldots, x_k, x_{k+1}) \pmod{N}$ to the congruence $a_1x_1 + \cdots + a_kx_k + a_{k+1}x_{k+1} \equiv c \pmod{N}$ is

$$\begin{aligned} & N^{k-1} \gcd(a_1, \ldots, a_k, N) \cdot \frac{N}{\gcd(a_1, \ldots, a_k, N)} \gcd(a_1, \ldots, a_{k+1}, N) \\ = & N^k \gcd(a_1, \ldots, a_{k+1}, N), \end{aligned}$$

which proves that the statement holds for $n = k + 1$. □

Theorem 2 *Let N be a composite number with smallest prime factor $p_{\min}$. Then, the advantage of the basic attack $\mathcal{A}_{\mathsf{basic}}$ against Poly-style PRFs (1) in the pseudorandomness experiment (defined in Definition 4) is $\frac{1}{p_{\min}} - \frac{1}{N}$.*

Proof. Assume that the advantage of algorithm $\mathcal{A}_{\mathsf{basic}}$ defined in the Definition 4 is $\varepsilon_{\mathsf{basic}}$. Then, it follows that

$$\varepsilon_{\mathsf{basic}} = 2\left(\Pr[b^* = b] - \frac{1}{2}\right) = \Pr[b^* = 0 \mid b = 0] + \Pr[b^* = 1 \mid b = 1] - 1.$$

Note that $\Pr[b^* = 0 \mid b = 0] = \Pr[\mathcal{O}^{\mathsf{PR}}_{0,\mathbf{k}}(x_1) \neq \mathcal{O}^{\mathsf{PR}}_{0,\mathbf{k}}(x_2)] = \frac{N-1}{N}$, and

$$\Pr[b^* = 1 \mid b = 1] = \Pr[\mathcal{O}^{\mathsf{PR}}_{1,\mathbf{k}}(x_1) = \mathcal{O}^{\mathsf{PR}}_{1,\mathbf{k}}(x_2) \mid \mathbf{k} \xleftarrow{\$} \mathbb{Z}_N^{d+1}].$$

By Lemma 1, we obtain

$$\begin{aligned} & \Pr[\mathcal{O}^{\mathsf{PR}}_{1,\mathbf{k}}(x_1) = \mathcal{O}^{\mathsf{PR}}_{1,\mathbf{k}}(x_2) \mid \mathbf{k} = (a_d, \ldots, a_1, a_0) \xleftarrow{\$} \mathbb{Z}_N^{d+1}] \\ = & \Pr[(x_1^d - x_2^d)a_d + \cdots + (x_1^2 - x_2^2)a_2 + (x_1 - x_2)a_1 \equiv 0 \pmod{N} \mid (a_d, \ldots, a_1) \xleftarrow{\$} \mathbb{Z}_N^d] \\ = & \frac{N^{d-1} \gcd(x_1^d - x_2^d, \ldots, x_1 - x_2, N)}{N^d} = \frac{\gcd(x_1 - x_2, N)}{N} = \frac{\frac{N}{p_{\min}} \gcd(i - j, p_{\min})}{N} = \frac{1}{p_{\min}}. \end{aligned}$$

Therefore, the advantage $\varepsilon_{\mathsf{basic}} = \frac{N-1}{N} + \frac{1}{p_{\min}} - 1 = \frac{1}{p_{\min}} - \frac{1}{N}$. □

The Enhanced Attack. One may make the following two observations.

(1) Our basic attack is analyzed under Definition 4, where the key $\mathbf{k}$ is sampled uniformly from the key space $\mathcal{K} = \mathbb{Z}_N^{d+1}$. A natural question is whether the basic attack can be avoided by choosing polynomials with certain structures or properties. In fact, regardless of the particular form or properties of the polynomials

from which f is sampled, the resulting construction can always be viewed as a degree-d Poly-style PRF with the key $\mathbf{k}$ sampled from a particular, possibly non-uniform, distribution over $\mathcal{K} = \mathbb{Z}_N^{d+1}$.

To this end, let $\mathcal{D}_{\mathcal{K}} = \{\mathcal{D}_{\mathcal{K},\lambda}\}_{\lambda\in\mathbb{N}^+}$ be a publicly specified and efficiently samplable distribution ensemble, where $\mathcal{D}_{\mathcal{K},\lambda}$ is a distribution over $\mathcal{K}_\lambda$. We define the $\mathcal{D}_{\mathcal{K}}$-pseudorandomness experiment identically to the standard pseudorandomness experiment in Definition 4, except that the key k is sampled according to $\mathcal{D}_{\mathcal{K},\lambda}$ rather than uniformly from $\mathcal{K}_\lambda$. Specifically, the advantage is defined as

$$\mathbf{Adv}^{\mathcal{D}_{\mathcal{K}}\text{-}\mathsf{PR}}_{\mathcal{A},\mathcal{F}}(\lambda) := 2\left(\Pr\left[b^* = b \;\middle|\; \begin{array}{c} b \xleftarrow{\$} \{0,1\},\ k \leftarrow \mathcal{D}_{\mathcal{K},\lambda},\ \mathsf{R} \xleftarrow{\$} \mathrm{Func}(\mathbb{Z}_N, \mathcal{E}), \\ b^* \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{PR}}_{b,k}}(1^\lambda, \mathsf{pp}) \end{array}\right] - \frac{1}{2}\right),$$

where pp encodes the public group-action parameters and the distribution ensemble $\mathcal{D}_{\mathcal{K}}$, and the oracles $\mathcal{O}^{\mathsf{PR}}_{b,k}$ $(b = 0, 1)$ are defined as in Definition 4 (here, we omit the security parameter λ from the subscripts). Correspondingly, we say that a family of keyed functions is $\mathcal{D}_{\mathcal{K}}$-pseudorandom if, for every PPT adversary $\mathcal{A}$, the advantage $\mathbf{Adv}^{\mathcal{D}_{\mathcal{K}}\text{-}\mathsf{PR}}_{\mathcal{A},\mathcal{F}}(\lambda)$ is negligible in λ. Thus, analyzing the pseudorandomness of Poly-PRFs with polynomials of specific forms is equivalent to analyzing corresponding $\mathcal{D}_{\mathcal{K}}$-pseudorandomness[7].

(2) Although the basic attack has only constant complexity (two oracle queries together with one equality test), its advantage $1/p_{\min}$ depends on the smallest prime factor of N. Can we construct an attack that succeeds with probability close to 1?

Our enhanced attack, presented below, answers both questions above. By combining the basic attack with the baby-step giant-step technique, we obtain an enhanced attack $\mathcal{A}_{\mathsf{enh}}$ that makes only two oracle queries and performs $O(\sqrt{p_{\min}})$ group-action computations. For any distribution $\mathcal{D}_{\mathcal{K}}$ over $\mathcal{K}$, the attack achieves an advantage close to 1 in the corresponding $\mathcal{D}_{\mathcal{K}}$-pseudorandomness experiment. This implies that, when N is composite, using polynomials of different forms cannot resist the enhanced attack and thus does not provide substantial improvement in the security level of Poly-style PRFs[8].

Our enhanced attack $\mathcal{A}_{\mathsf{enh}}$ proceeds as follows.

1. $\mathcal{A}_{\mathsf{enh}}$ runs Steps 1 and 2 of $\mathcal{A}_{\mathsf{basic}}$ and obtains two oracle responses Y_1 and Y_2. If $Y_1 = Y_2$, then $\mathcal{A}_{\mathsf{enh}}$ outputs the guess $b^* = 1$; otherwise, it proceeds to Step 2. (Here, the notation for all other values follows that used in $\mathcal{A}_{\mathsf{basic}}$.)
2. Let $m = \lceil\sqrt{p_{\min}}\rceil$. For each $s, t \in \{0, 1, \ldots, m-1\}$, $\mathcal{A}_{\mathsf{enh}}$ computes

$$U_s = \left(-s\frac{N}{p_{\min}}\right) \diamond Y_1, \qquad V_t = \left(tm\frac{N}{p_{\min}}\right) \diamond Y_2.$$

[7] Note that $\mathcal{D}_{\mathcal{K}}$-pseudorandomness implies that $\max_{k\in\mathcal{K}} \Pr[k \leftarrow \mathcal{D}_{\mathcal{K},\lambda}]$ is negligible in λ.

[8] De Saint Guilhem [42] and Levin [32] each discussed in their respective works the use of polynomials with specific structures to ensure the security of their schemes. For example, one of the polynomials they considered is of the form $f(x) = a(x+b)^3+c = ax^3 + 3abx^2 + 3ab^2x + (ab^3 + c)$, where $a \neq 0$. This polynomial corresponds to the coefficient vector $\mathbf{k} = (ab^3 + c, 3ab^2, 3ab, a) \in \mathbb{Z}_N^4$. Therefore, the distribution of $\mathbf{k}$ can be viewed as a distribution over $\mathbb{Z}_N^4$ induced by the distribution of (a, b, c).

3. $\mathcal{A}_{\mathsf{enh}}$ returns $b^* = 1$ if there exists a pair $(\hat{s}, \hat{t})$ such that $U_{\hat{s}} = V_{\hat{t}}$; otherwise, it returns $b^* = 0$.

Theorem 3 *Let N be a composite number with smallest prime factor $p_{\min}$. Then, for any distribution $\mathcal{D}_{\mathcal{K}}$ over the key space $\mathcal{K} = \mathbb{Z}_N^{d+1}$, the advantage of the enhanced attack $\mathcal{A}_{\mathsf{enh}}$ against Poly-style PRFs (1) in the $\mathcal{D}_{\mathcal{K}}$-pseudorandomness experiment is $1 - \frac{p_{\min}}{N}$, which is at least $1 - \frac{1}{\sqrt{N}}$.*

Proof. Suppose that, for $\delta = 1, 2$, $Y_\delta = y_\delta \diamond E_0$. Here, we denote the canonical representatives of $y_1, y_2 \in \mathbb{Z}_N$ in $\{0, \ldots, N-1\}$ by $y_1^{(N)}$ and $y_2^{(N)}$, respectively, and assume that

$$y_1^{(N)} = z_1 \frac{N}{p_{\min}} + r_1, \ y_2^{(N)} = z_2 \frac{N}{p_{\min}} + r_2,$$

where $z_1, z_2 \in \{0, \ldots, p_{\min} - 1\}$, and $r_1, r_2 \in \{0, \ldots, \frac{N}{p_{\min}} - 1\}$. Next, we show that $\mathcal{A}_{\mathsf{enh}}$ outputs $b^* = 1$, namely, that there exists a pair $(\hat{s}, \hat{t})$ such that $U_{\hat{s}} = V_{\hat{t}}$, if and only if $y_1^{(N)} \equiv y_2^{(N)} \pmod{\frac{N}{p_{\min}}}$.

The former statement is equivalent to saying that there exists a pair $(\hat{s}, \hat{t}) \in \{0, 1, \ldots, m-1\}^2$ such that

$$\left(z_1 \frac{N}{p_{\min}} + r_1\right) - \hat{s} \frac{N}{p_{\min}} \equiv \left(z_2 \frac{N}{p_{\min}} + r_2\right) + \hat{t}m \frac{N}{p_{\min}} \pmod{N}.$$

Equivalently,

$$\left((z_1 - z_2) \frac{N}{p_{\min}} + (r_1 - r_2)\right) \equiv \left((\hat{t}m + \hat{s}) \frac{N}{p_{\min}}\right) \pmod{N} \tag{3}$$

It follows from the above congruence that $r_1 - r_2 \equiv 0 \pmod{\frac{N}{p_{\min}}}$. Note that $r_1, r_2 \in \{0, \ldots, \frac{N}{p_{\min}} - 1\}$. Therefore, the above congruence (3) implies $r_1 = r_2$, and thus $y_1^{(N)} \equiv y_2^{(N)} \pmod{\frac{N}{p_{\min}}}$.

Conversely, if $y_1^{(N)} \equiv y_2^{(N)} \pmod{\frac{N}{p_{\min}}}$, then we have $r_1 = r_2$. Observe that the set $\{tm + s \mid t, s \in \{0, 1, \ldots, m-1\}\}$ contains a complete residue system modulo $p_{\min}$, which means that for any z_1, z_2, there exists a pair $(\hat{s}, \hat{t}) \in \{0, 1, \ldots, m-1\}^2$ such that

$$z_1 - z_2 \equiv \hat{t}m + \hat{s} \pmod{p_{\min}} \tag{4}$$

Multiplying both sides of the congruence (4), as well as the modulus, by $\frac{N}{p_{\min}}$, we obtain the congruence (3), and thus $U_{\hat{s}} = V_{\hat{t}}$.

Using the above equivalence, we can easily compute the advantage of $\mathcal{A}_{\mathsf{enh}}$ for $\mathcal{D}_{\mathcal{K}}$-pseudorandomness experiment, denoted by $\varepsilon_{\mathsf{enh}}$. Here, we regard evaluation key $\mathbf{k}$ as a random variable distributed according to $\mathcal{D}_{\mathcal{K}}$, and use $\hat{\mathbf{k}} \in \mathrm{supp}(\mathcal{D}_{\mathcal{K}})$

to denote a particular realization of $\mathbf{k}$. First, we have

$$\begin{aligned}\varepsilon_{\mathsf{enh}} &= 2\left(\Pr[b^* = b] - \frac{1}{2}\right) \\ &= 2\left(\sum_{\hat{\mathbf{k}} \in \mathrm{supp}(\mathcal{D}_{\mathcal{K}})} \Pr[b^* = b \mid \mathbf{k} = \hat{\mathbf{k}}] \Pr[\mathbf{k} = \hat{\mathbf{k}}] - \frac{1}{2}\right). \end{aligned} \tag{5}$$

and

$$\Pr[b^* = b \mid \mathbf{k} = \hat{\mathbf{k}}] = \frac{1}{2}\left(\Pr[b^* = 0 \mid \mathbf{k} = \hat{\mathbf{k}}, b = 0] + \Pr[b^* = 1 \mid \mathbf{k} = \hat{\mathbf{k}}, b = 1]\right) \tag{6}$$

On the one hand, we have

$$\begin{aligned}&\Pr[b^* = 0 \mid \mathbf{k} = \hat{\mathbf{k}}, b = 0] \\ =&\Pr\left[y_1^{(N)} \not\equiv y_2^{(N)} \pmod{\frac{N}{p_{\min}}} \;\middle|\; y_1^{(N)}, y_2^{(N)} \xleftarrow{\$} \{0, 1, \ldots, N-1\}\right] \\ =&1 - \frac{p_{\min}}{N}. \end{aligned} \tag{7}$$

On the other hand, when $b = 1$, for every $\hat{\mathbf{k}} = (a_0, \ldots, a_d) \in \mathrm{supp}(\mathcal{D}_{\mathcal{K}})$, the following congruence holds:

$$y_1^{(N)} - y_2^{(N)} \equiv (i-j)\frac{N}{p_{\min}}\left(a_1 + a_2(x_1 + x_2) + \ldots a_d(\sum_{\alpha=0}^{d-1} x_1^{d-1-\alpha} x_2^{\alpha})\right) \equiv 0 \pmod{\frac{N}{p_{\min}}},$$

and thus $\mathcal{A}_{\mathsf{enh}}$ always returns $b^* = 1$. Therefore,

$$\Pr[b^* = 1 \mid \mathbf{k} = \hat{\mathbf{k}}, b = 1] = 1. \tag{8}$$

Combining Equations (6–8), we obtain

$$\Pr[b^* = b \mid \mathbf{k} = \hat{\mathbf{k}}] = 1 - \frac{p_{\min}}{2N}.$$

Further combining the above equation with Equation (5), and observing that $\sum_{\hat{\mathbf{k}} \in \mathrm{supp}(\mathcal{D}_{\mathcal{K}})} \Pr[\mathbf{k} = \hat{\mathbf{k}}] = 1$, we obtain

$$\varepsilon_{\mathsf{enh}} = 2\left(\left(1 - \frac{p_{\min}}{2N}\right) - \frac{1}{2}\right) = 1 - \frac{p_{\min}}{N} \geqslant 1 - \frac{1}{\sqrt{N}},$$

which completes the proof. □

Attack on Levin's VRF.[9] Levin and Pedersen [32] adopt the following structure in their construction of VRFs using Poly-style PRFs. On input x, the evaluator first computes $E_x = f(x) \diamond E_0$ and then generates a non-interactive zero-knowledge proof π for the correctness of E_x with respect to the public commitment pk to the secret polynomial f. Finally, it computes $h = \mathcal{H}(\mathsf{pk}, x, E_x)$. Here,

[9] The attack against Levin and Pedersen's construction presented in the concurrent work of Lai, Yu, and Zhou [31] is identical to the attack described here. This attack is

$\mathcal{H} : \{0,1\}^* \to \{0,1\}^{2\lambda}$ be a hash function, and h is the final VRF output, while (E_x, π) constitutes the proof. The verifier accepts if and only if $h = \mathcal{H}(\mathsf{pk}, x, E_x)$ and π is a valid proof for the statement (pk, x, E_x). Note that although the final VRF output is a hash value, the underlying Poly-style PRF value $E_x = f(x) \diamond E_0$ is still revealed to the adversary as part of the proof. Therefore, our enhanced attack can be slightly modified as follows. Let $p_{\min}$ be the smallest prime factor of N and set $t = N/p_{\min}$. To break the residual pseudorandomness (see definition in Section 2.2), the adversary first queries the evaluation oracle at x and obtains $(h, (E_x, \pi))$, and then chooses the fresh challenge input $\hat{x} = x + t$. Upon receiving the challenge value $\hat{h}$, the adversary checks whether there exists some $j \in \{0, 1, \ldots, p_{\min} - 1\}$ such that $\hat{h} = \mathcal{H}(\mathsf{pk}, \hat{x}, (tj) \diamond E_x)$, and guesses that $\hat{h}$ is a pseudorandom value if and only if such a j exists, and that $\hat{h}$ is a truly random value otherwise. This attack requires $O(p_{\min})$ classical group-action and hash evaluations and achieves advantage close to 1. By evaluating the hash function and the group action in quantum superposition, a quantum adversary can apply Grover's algorithm, reducing the attack complexity to $O(\sqrt{p_{\min}})$.

Comparison with Lai's Attack [29]. Lai proposed a key-recovery attack against Poly-style PRFs in [29]. Lai's attack assumes that the prime factorization of N is known and that N is square-free. Let $N = \prod_{i=1}^{s} p_i$, where $p_1, \ldots, p_s$ are distinct primes, and set $t_i = N/p_i$. For each prime factor p_i, the attack queries the PRF evaluation oracle $\mathcal{O}_k^{\mathsf{PR}''}$ from Equivalent Definition 2 (see Section 2.2) on pairs of inputs whose difference is t_i. Since $\mathcal{O}_k^{\mathsf{PR}''}(x + t_i) = (f(x + t_i) - f(x)) \diamond \mathcal{O}_k^{\mathsf{PR}''}(x)$ and $t_i \mid f(x + t_i) - f(x)$, to get the value of $f(x + t_i) - f(x)$, one only needs to solve the GA-DLog instance $(\mathcal{O}_k^{\mathsf{PR}''}(x), \mathcal{O}_k^{\mathsf{PR}''}(x + t_i))$ over the order-p_i subgroup $\langle t_i \rangle = \{t_i z \bmod N \mid z \in \mathbb{Z}_N\} \subseteq \mathbb{Z}_N$. By solving GA-DLog in this subgroup for sufficiently many such input pairs, the adversary obtains a system of linear equations whose solution yields the nonconstant coefficients $(a_1, \ldots, a_d)$ of the polynomial modulo p_i. Repeating this procedure for all prime factors of N and applying the Chinese remainder theorem, the attack recovers the nonconstant coefficients $(a_1, \ldots, a_d)$ of the polynomial over $\mathbb{Z}_N$. It can then compute the PRF output at any input x: by querying $\mathcal{O}_k^{\mathsf{PR}''}$ at 0, it obtains $\mathcal{O}_k^{\mathsf{PR}''}(0) = a_0 \diamond E_0$, and subsequently computes $\mathcal{O}_k^{\mathsf{PR}''}(x) = (a_d x^d + \cdots + a_1 x) \diamond \mathcal{O}_k^{\mathsf{PR}''}(0)$.

a variant of our enhanced attack. The main difference stems from the use of the hash function $\mathcal{H}$ in Levin's construction to compute the final PRF value. Consequently, in the classical setting, the adversary must exhaustively test the $p_{\min}$ candidate elements, and the BSGS technique can no longer be used to reduce the complexity to $O(\sqrt{p_{\min}})$. Following the attacks of Lai, Yu, and Zhou [31], Levin and Pedersen have recently revised their parameter selection. Following the subgroup-based approach of [39], they use the SCALLOP framework to generate an ideal class group of order $N = 2p$, where p is a large prime. The underlying group action remains defined over $\mathbb{Z}_N$, while the input space is restricted to its order-p subgroup. This particular parameter choice is secure according to our analysis in the GGAM$^{\mathsf{T}}$ model in Section 4. Nevertheless, we will show that this subgroup-based approach lacks generality and strongly depends on the factorization structure of N.

Clearly, Lai's algorithm must solve several GA-DLog instances in every prime-order subgroup of $\mathbb{Z}_N$. Therefore, its overall complexity is dominated by the largest prime factor $p_{\max}$ of N. In contrast, our enhanced attack requires two oracle queries and only $O(\sqrt{p_{\min}})$ group-action evaluations and achieves advantage close to 1. This implies that, under most parameter settings considered in current research, Poly-style PRFs can be broken very efficiently, rather than merely having a reduced effective security level, as suggested by Lai in his note [29]. For example, in CSIDH-512, N has small prime factors such as 3 and 37, which are much smaller than $p_{\max}$, a 135-bit prime. Moreover, Levin and Pedersen [32] recommend constructing class groups of order $c-1$ via the SCALLOP method, where $c = 2^{257} + 269451$ is prime, so that $c-1$ factors as 2 times a 256-bit prime. We believe that the discrepancy between our attack and Lai's stems from the fact that Lai's attack breaks pseudorandomness by recovering the secret key, thereby exceeding what is required to break pseudorandomness. However, it is possible that some applications require only the one-more unpredictability of Poly-style PRFs (i.e., an adversary is allowed to adaptively query the evaluation oracle on polynomially many inputs and then attempts to correctly compute the PRF output at a fresh point). In such cases, Lai's key-recovery attack remains relevant. Table 2 presents an intuitive comparison among our basic attack, our enhanced attack, and Lai's attack. In the next section, we will show how to construct a new family of PRFs from group actions that is simultaneously provably secure for arbitrary composite N and supports efficient proofs of correctness.

	Lai's attack[29]	**Our attack (basic)**	**Our attack (enhanced)**
Complexity	Solving GA-DLog with subgroup of order $p_{\max}$	3 (constant)	$O(\sqrt{p_{\min}})$
Advantage	1	$1/p_{\min} - \mathrm{negl}(\lambda)$	$1 - \mathrm{negl}(\lambda)$

Remark: Here, λ denotes the security parameter, defined as $\lceil \log_2 N \rceil$.

Table 2. Comparison between Lai's attack and our attacks.

4 The Proposed PRFs

In this section, we present our core technique, *Restrict-then-Expand*, to construct a new PRF family. The section is organized into three components: restriction, optimization and expansion.

4.1 Restricting the Input Space

It can be observed that both our attacks and Lai's attack rely on high-gap value queries to the PRF evaluation oracle. Therefore, restricting the input space is the first step toward constructing our final PRFs. In this section, we restrict the valid input space of Poly-style PRFs to a subset $\mathcal{S} \subseteq \mathbb{Z}_N$ and analyze their security in the Generic Group Action Model with Twists (GGAM$^{\mathsf{T}}$), introduced by Duman

et al. [19]. This model is a variant of the Generic Group Model (GGM) [45] tailored to isogeny-based group actions.

Generic Group Action Model with Twists ($\mathsf{GGAM}^{\mathsf{T}}$) [19]. Given a $\mathrm{CEGA}^{\mathsf{T}} = (\mathbb{Z}_N, \mathcal{X}, \diamond, X_0)$, let $\sigma_{\mathcal{X}} : \mathcal{X} \to \{0,1\}^{l(\lambda)}$ be a uniformly random injective labeling function, and define the image of $\mathcal{X}$ under $\sigma_{\mathcal{X}}$ as the set of labels $\Sigma := \{\sigma_{\mathcal{X}}(X) \mid X \in \mathcal{X}\}$. For every $X \in \mathcal{X}$, the element $\sigma_{\mathcal{X}}(X) \in \Sigma$ is called the label of X. The function $\sigma_{\mathcal{X}}$ can be viewed as assigning to each element of $\mathcal{X}$ a random label without any algebraic meaning. Consequently, the labels themselves reveal no information beyond whether their underlying elements are equal. Moreover, we assume that the length of labels $l(\lambda)$ is sufficiently large to ensure that randomly guessing a label having a corresponding preimage in $\mathcal{X}$ is hard. A *generic group action algorithm with twists* is an algorithm whose initial inputs in $\mathcal{X}$ are replaced by their corresponding labels in Σ, and which performs the group action and twist operations via the oracles $\mathcal{O}^{\mathsf{GA}} : \mathbb{Z}_N \times \Sigma \to \Sigma$ and $\mathcal{O}^{\mathsf{Tw}} : \Sigma \to \Sigma$, respectively. The oracle $\mathcal{O}^{\mathsf{GA}}$ takes as input $a \in \mathbb{Z}_N$ and a label $\sigma_{\mathcal{X}}(X)$ for $X \in \mathcal{X}$, and outputs $\sigma_{\mathcal{X}}(a \diamond X)$; the oracle $\mathcal{O}^{\mathsf{Tw}}$ takes as input a label $\sigma_{\mathcal{X}}(X)$, where $X = a \diamond X_0$, and outputs $\sigma_{\mathcal{X}}((-a) \diamond X_0)$. The $\mathrm{GGAM}^{\mathsf{T}}$ is a model where the adversary is modelled as a generic group action algorithm with twists.

Poly-style PRFs with Restricted Input Space. Since both our attacks and Lai's attack[10] are generic group action algorithms captured by the $\mathrm{GGAM}^{\mathsf{T}}$, establishing security in this model implies resistance against all generic algorithms captured by this model, including these attacks. Therefore, we present the following Theorem 4, which characterizes the security of Poly-style PRFs in the $\mathrm{GGAM}^{\mathsf{T}}$ under a restricted input space $\mathcal{S} \subseteq \mathbb{Z}_N$. The upper bound in the theorem implies that, by choosing an appropriate $\mathcal{S}$, when the adversary $\mathcal{A}$ makes only polynomially many queries, including queries to the evaluation function in the pseudorandomness experiment and queries to the group-action and twist oracles, the distinguishing advantage of $\mathcal{A}$ is negligible, and hence the Poly-style PRFs are secure in the $\mathrm{GGAM}^{\mathsf{T}}$.

Theorem 4 *Consider a degree-d Poly-style PRF $F^{\mathsf{poly}}(\mathbf{k}, x) = (a_d x^d + \cdots + a_1 x + a_0) \diamond E_0$ defined over a $\mathrm{CEGA}^{\mathsf{T}}(\mathbb{Z}_N, \mathcal{E}, \diamond, E_0)$, where the secret key $\mathbf{k} = (a_0, a_1, \ldots, a_d) \xleftarrow{\$} \mathbb{Z}_N^{d+1}$. For a fixed subset $\mathcal{S} \subseteq \mathbb{Z}_N$ with $|\mathcal{S}| \geqslant 2$, let the truly random function $\mathsf{R} \xleftarrow{\$} \mathrm{Func}(\mathcal{S}, \mathcal{E})$. For such a subset $\mathcal{S}$, a key $\mathbf{k} \in \mathbb{Z}_N^{d+1}$, and a bit $b \in \{0,1\}$, we define the oracle $\mathcal{O}^{\mathcal{S}}_{b,\mathbf{k}}$ as follows.*

Input $x \in \mathcal{S}$; $Y_0 \leftarrow \mathsf{R}(x)$, $Y_1 \leftarrow F^{\mathsf{poly}}(\mathbf{k}, x)$; Return $(\sigma_{\mathcal{E}}(Y_b), \sigma_{\mathcal{E}}(Y_{1-b}))$ | **Oracle** $\mathcal{O}^{\mathcal{S}}_{b,\mathbf{k}}(\cdot)$

Let $\mathcal{A}$ be a generic group action algorithm that outputs a bit b^* after making q queries to the oracle $\mathcal{O}^{\mathcal{S}}_{b,\mathbf{k}}$, and a total of q' queries to the group action oracle

[10] Strictly speaking, Lai's attack equipped with a generic algorithm for solving the subgroup GA-DLog problem is captured by the $\mathrm{GGAM}^{\mathsf{T}}$, provided that the subgroup GA-DLog subroutine makes only classical queries to the group-action oracle.

$\mathcal{O}^{\mathsf{GA}}$ *and the twist oracle* $\mathcal{O}^{\mathsf{Tw}}$. *Let* $\varepsilon_{\mathcal{A}}$ *denote the advantage of* $\mathcal{A}$, *defined as*

$$\varepsilon_{\mathcal{A}} := 2\left(\Pr[b^* = b \mid b \xleftarrow{\$} \{0,1\},\ \mathbf{k} \xleftarrow{\$} \mathbb{Z}_N^{d+1},\ b^* \leftarrow \mathcal{A}^{\mathcal{O}^{\mathcal{S}}_{b,\mathbf{k}},\mathcal{O}^{\mathsf{GA}},\mathcal{O}^{\mathsf{Tw}}}(\mathbb{Z}_N, \sigma_{\mathcal{E}}(E_0), d, \mathcal{S})] - \frac{1}{2}\right).$$

Then, for any $q, q' \in \mathbb{N}$ *such that* $2q+q' \leqslant \sqrt{\frac{N}{D_{\mathcal{S}}}} - 1$, *the advantage of* $\mathcal{A}$ *satisfies*

$$\varepsilon_{\mathcal{A}} < \frac{3(2q+q'+1)(2q+q')D_{\mathcal{S}}}{N}, \quad \textit{and hence} \quad \varepsilon_{\mathcal{A}} = O\left(\frac{(2q+q')^2 D_{\mathcal{S}}}{N}\right).$$

where $D_{\mathcal{S}} = \max\left\{\gcd(2,N), \max_{\mathrm{x},\mathrm{x}' \in \mathcal{S},\ \mathrm{x} \neq \mathrm{x}'} \gcd(\mathrm{x}-\mathrm{x}', N)\right\}$.

Proof. The proof uses an argument similar to that of Duman et al. [19], which clearly conveys the main idea of analysis in GGAM$^{\mathsf{T}}$. In summary, a generic group action algorithm initially has a random label σ_0 corresponding to E_0. It needs to obtain additional labels by querying the oracles $\mathcal{O}^{\mathcal{S}}_{b,\mathbf{k}}$, $\mathcal{O}^{\mathsf{GA}}$, and $\mathcal{O}^{\mathsf{Tw}}$, and extract useful information by checking whether collisions occur among these labels. Nevertheless, readers may also refer to Appendix B additional details for the proof.

Assume that $\mathcal{A}$ makes q queries to the oracle $\mathcal{O}^{\mathcal{S}}_{b,\mathbf{k}}$ on inputs $\mathrm{x}_1, \ldots, \mathrm{x}_q \in \mathcal{S}$. Without loss of generality, we assume $\mathrm{x}_1, \ldots, \mathrm{x}_q$ are pairwise distinct. We denote the oracle output on input x_i ($i \in [q]$) by $(\sigma_{\mathcal{E}}(Y_b^{(i)}), \sigma_{\mathcal{E}}(Y_{1-b}^{(i)}))$. For every $i \in [q]$, we define indeterminates U_i and V_i that represent the values $\sigma_{\mathcal{E}}(Y_b^{(i)})$ and $\sigma_{\mathcal{E}}(Y_{1-b}^{(i)})$, respectively. By making a total of q' queries to the oracles $\mathcal{O}^{\mathsf{GA}}$ and $\mathcal{O}^{\mathsf{Tw}}$, the adversary $\mathcal{A}$ ultimately obtains $2q+q'$ polynomials $\{f_i(U_1, \ldots, U_q, V_1, \ldots, V_q)\}_{i=1}^{2q+q'}$ over $\mathbb{Z}_N[U_1, \ldots, U_q, V_1, \ldots, V_q]$ (in this proof, we assume that these $2q+q'$ polynomials are pairwise distinct[11]), each of which can only take one of the following forms:

$$c, \qquad \pm U_i + c \ \ (i \in [q]), \qquad \text{and} \qquad \pm V_i + c \ \ (i \in [q]),$$

where $c \in \mathbb{Z}_N$ is a constant known to $\mathcal{A}$. At the end of the game, (U_i, V_i) is set to either $(\mathrm{x}_i^d A_d + \cdots + \mathrm{x}_i A_1 + A_0, R_i)$ or $(R_i, \mathrm{x}_i^d A_d + \cdots + \mathrm{x}_i A_1 + A_0)$, where $A_0, \ldots, A_d$ and R_i ($i \in [q]$) are indeterminates. We define

$$\begin{aligned}
&\hat{f}_i^{(0)}(A_0, \ldots, A_d, R_1, \ldots, R_q) \\
&= f_i(R_1, \ldots, R_q, \mathrm{x}_1^d A_d + \cdots + \mathrm{x}_1 A_1 + A_0, \ldots, \mathrm{x}_q^d A_d + \cdots + \mathrm{x}_q A_1 + A_0), \\
&\hat{f}_i^{(1)}(A_0, \ldots, A_d, R_1, \ldots, R_q) \\
&= f_i(\mathrm{x}_1^d A_d + \cdots + \mathrm{x}_1 A_1 + A_0, \ldots, \mathrm{x}_q^d A_d + \cdots + \mathrm{x}_q A_1 + A_0, R_1, \ldots, R_q),
\end{aligned}$$

and obtain $2(2q+q')$ polynomials $\{\hat{f}_i^{(b)}\}_{b \in \{0,1\}, i \in [2q+q']}$ over $\mathbb{Z}_N[A_0, \ldots, A_d, R_1, \ldots, R_q]$. Moreover, for each $b \in \{0,1\}$, we define $\hat{f}_0^{(b)} := 0$ to denote the

[11] These assumptions, including the earlier assumption that $\mathrm{x}_1, \ldots, \mathrm{x}_q$ are pairwise distinct, are justified because repeated queries or repeated polynomials provide only redundant information to $\mathcal{A}$, which gives $\mathcal{A}$ no additional advantage.

zero polynomial associated with the label $\sigma_{\mathcal{E}}(E_0)$. Let $\mathcal{Z}(f)$ denote the set of zeros of the polynomial f over $\mathbb{Z}_N$, we define the following subset of $\mathbb{Z}_N^{d+q+1}$:

$$\Psi_b := \bigcup_{0 \leqslant i < j \leqslant 2q+q'} \mathcal{Z}\left(\hat{f}_i^{(b)} - \hat{f}_j^{(b)}\right).$$

and let Coll denote the event that there exists at least one collision among the $2q+q'+1$ labels (note that the label $\sigma_{\mathcal{E}}(E_0)$ is also included here) obtained by $\mathcal{A}$. To upper-bound the probability of the event Coll, our goal is to upper-bound the probability that a randomly chosen $(A_0, \ldots, A_d, R_1, \ldots, R_q) \in \mathbb{Z}_N^{d+q+1}$ belongs to the set Ψ_b. To this end, it suffices to upper-bound the number of solutions of the polynomial $\hat{f}_i^{(b)} - \hat{f}_j^{(b)}$, where $0 \leqslant i < j \leqslant 2q + q'$. Note that, regardless of whether $b = 0$ or $b = 1$, and up to a sign, the polynomial $\hat{f}_i^{(b)} - \hat{f}_j^{(b)}$ can only take one of the following eight forms:

$$\begin{aligned}
&(1)\ A_0 + \mathrm{x}A_1 + \cdots + \mathrm{x}^d A_d + c, \quad (2)\ R_i + c,\\
&(3)\ (\mathrm{x} - \mathrm{x}')A_1 + \cdots + (\mathrm{x}^d - \mathrm{x}'^d)A_d + c \quad (\mathrm{x} \neq \mathrm{x}'),\\
&(4)\ 2A_0 + (\mathrm{x} + \mathrm{x}')A_1 + \cdots + (\mathrm{x}^d + \mathrm{x}'^d)A_d + c,\\
&(5)\ A_0 + \mathrm{x}A_1 + \cdots + \mathrm{x}^d A_d \pm R_i + c,\\
&(6)\ R_i \pm R_j + c \ \ (i \neq j), \quad (7)\ 2R_i + c, \quad (8)\ c \quad (c \neq 0).
\end{aligned}$$

where $\mathrm{x}, \mathrm{x}' \in \{\mathrm{x}_i\}_{i=1}^q$, $c \in \mathbb{Z}_N$ is a constant. Based on Lemma 1, we bound the number of root $(A_0, \ldots, A_d, R_1, \ldots, R_q) \in \mathbb{Z}_N^{d+q+1}$ of each type of polynomial. Among them, the polynomial (8) has no solutions; the polynomials (1), (2), (5) and (6) each have exactly N^{d+q} solutions; the polynomials (4) and (7) each have at most $N^{d+q}\gcd(2, N)$ solutions; the polynomial (3) has at most

$$N^{d+q} \max_{\mathrm{x},\mathrm{x}' \in \mathcal{S},\ \mathrm{x} \neq \mathrm{x}'} \gcd(\mathrm{x} - \mathrm{x}', N)$$

solutions, and thus can potentially admit the largest number of solutions among the eight cases. Let

$$D_{\mathcal{S}} := \max\{\gcd(2, N), \max_{\mathrm{x},\mathrm{x}' \in \mathcal{S},\ \mathrm{x} \neq \mathrm{x}'} \gcd(\mathrm{x} - \mathrm{x}', N)\}.$$

We obtain the upper bound $|\mathcal{Z}(\hat{f}_i^{(b)} - \hat{f}_j^{(b)})| \leqslant N^{d+q} D_{\mathcal{S}}$. Therefore, after $\mathcal{A}$ has finished its queries, the probability that a uniformly sampled $(A_0, \ldots, A_d, R_1, \ldots, R_q) \xleftarrow{\$} \mathbb{Z}_N^{d+q+1}$ belongs to either Ψ_0 or Ψ_1, i.e., belongs to $\Psi_0 \cup \Psi_1$, is upper-bounded by

$$\frac{2 \cdot \binom{2q+q'+1}{2} N^{d+q} D_{\mathcal{S}}}{N^{d+q+1}} = \frac{(2q + q' + 1)(2q + q')D_{\mathcal{S}}}{N}.$$

Hence, $\Pr[\mathsf{coll}] \leqslant (2q + q' + 1)(2q + q')D_{\mathcal{S}}/N$.

Next, we upper bound the probability that the adversary correctly guesses b conditioned on no collision occurring among the labels, i.e. $\Pr[\mathcal{A}\ wins \mid \overline{\mathsf{Coll}}]$[12]. Our main idea is to decompose the event $\overline{\mathsf{Coll}}$ according to all possible transcripts and establish a common upper bound for every fixed transcript. To this end, we regard the transcript, consisting of all queries issued by the adversary together with the corresponding response labels, as a random variable, denoted by $\mathbf{T}$. We denote a particular transcript by $\mathbf{t}$. Define $\mathsf{Supp} := \left\{\mathbf{t} \;\middle|\; \Pr[\mathbf{T}=\mathbf{t} \mid \overline{\mathsf{Coll}}] > 0\right\}$. Then, the probability we need to upper-bound can be written as

$$\Pr[\mathcal{A}\ wins \mid \overline{\mathsf{Coll}}] = \sum_{\mathbf{t}\in\mathsf{Supp}} \Pr[\mathcal{A}\ wins \mid \mathbf{T}=\mathbf{t}, \overline{\mathsf{Coll}}] \Pr[\mathbf{T}=\mathbf{t} \mid \overline{\mathsf{Coll}}]. \tag{9}$$

Observe that[13]

$$\begin{aligned}
&\Pr[\mathcal{A}\ wins \mid \mathbf{T}=\mathbf{t},\ \overline{\mathsf{Coll}}] = \Pr[\mathcal{A}\ wins \mid \mathbf{T}=\mathbf{t}] \\
&\leqslant \max\{\Pr[b=0 \mid \mathbf{T}=\mathbf{t}],\ \Pr[b=1 \mid \mathbf{T}=\mathbf{t}]\} \\
&= \frac{\max\limits_{\delta\in\{0,1\}}\{N^{d+q+1} - |\Psi_{\delta,\mathbf{t}}|\}}{(N^{d+q+1} - |\Psi_{0,\mathbf{t}}|) + (N^{d+q+1} - |\Psi_{1,\mathbf{t}}|)} \\
&\leqslant \frac{N^{d+q+1}}{2N^{d+q+1} - 2\binom{2q+q'+1}{2} N^{d+q} \cdot D_{\mathcal{S}}} = \frac{N}{2N - (2q+q'+1)(2q+q') \cdot D_{\mathcal{S}}}
\end{aligned} \tag{10}$$

where $\Psi_{\delta,\mathbf{t}}$ ($\delta \in \{0,1\}$) denotes the set Ψ_δ uniquely determined by the transcript $\mathbf{t}$. Let $Q = (2q+q'+1)(2q+q')$. Note that Inequality (10) above holds under the condition that $QD_{\mathcal{S}} < 2N$. Under the condition $2q+q' \leqslant \sqrt{\frac{N}{D_{\mathcal{S}}}} - 1$, we have $(2q+q'+1)(2q+q') < (2q+q'+1)^2 \leqslant \frac{N}{D_{\mathcal{S}}}$, which implies $QD_{\mathcal{S}} < N < 2N$.

Finally, combining (9) and (10), we obtain the upper bound

$$\Pr[\mathcal{A}\ wins \mid \overline{\mathsf{Coll}}] \leqslant \frac{N}{2N - QD_{\mathcal{S}}}\ .$$

It can be observed that, when the number of queries is polynomially bounded, this upper bound is close to $1/2$ only when $D_{\mathcal{S}}$ is sufficiently small. We can now upper-bound the probability that $\mathcal{A}$ wins the experiment as follows:

$$\begin{aligned}
&\Pr[\mathcal{A}\ wins] = \Pr[\mathcal{A}\ wins \mid \mathsf{Coll}] \Pr[\mathsf{Coll}] + \Pr[\mathcal{A}\ wins \mid \overline{\mathsf{Coll}}] \Pr[\overline{\mathsf{Coll}}] \\
&\leqslant \Pr[\mathsf{Coll}] + \Pr[\mathcal{A}\ wins \mid \overline{\mathsf{Coll}}] \leqslant \frac{QD_{\mathcal{S}}}{N} + \frac{N}{2N - QD_{\mathcal{S}}}.
\end{aligned}$$

Therefore, the advantage of $\mathcal{A}$ in the experiment satisfies

$$\varepsilon_{\mathcal{A}} = 2\left(\Pr[\mathcal{A}\ wins] - \frac{1}{2}\right) \leqslant \frac{2QD_{\mathcal{S}}}{N} + \frac{QD_{\mathcal{S}}}{2N - QD_{\mathcal{S}}}$$

[12] This probability may be intuitively regarded as being close to $1/2$. In fact, if $\mathcal{S}$ is not chosen appropriately, this probability can even be 1. Our enhanced attack provides such an example (note that in our enhanced attack, if no collision occurs, the challenge oracle must be a truly random function).

[13] A more detailed derivation for Inequality (10) is provided in Appendix B.

Again, by the condition $2q + q' \leqslant \sqrt{\frac{N}{D_{\mathcal{S}}}} - 1$, we have $QD_{\mathcal{S}} \leqslant N$, and hence

$$\frac{QD_{\mathcal{S}}}{2N - QD_{\mathcal{S}}} < \frac{QD_{\mathcal{S}}}{N}.$$

Therefore,

$$\varepsilon_{\mathcal{A}} < \frac{3QD_{\mathcal{S}}}{N} = \frac{3(2q + q' + 1)(2q + q')D_{\mathcal{S}}}{N},$$

which completes the proof of the upper bound in Theorem 4. □

From the above Theorem 4, if we choose an appropriate subset $\mathcal{S} \subseteq \mathbb{Z}_N$ such that $D_{\mathcal{S}}$ is sufficiently small, for example, $D_{\mathcal{S}} \leqslant \mathrm{poly}(\lambda)$, then the above upper bound $O((2q + q')^2)D_{\mathcal{S}}/N$ becomes comparable to that of the GA-DLog problem, namely $2q^2/N$ (Theorem 2, [19]). Moreover, we emphasize that the condition $2q+q' \leqslant \sqrt{N/D_{\mathcal{S}}}-1$ does not affect the security guarantee provided by our theorem. Actually, if $2q+q' \geqslant \sqrt{N/D_{\mathcal{S}}}$, the upper bound already approaches a constant. Indeed, choose distinct $x_1, x_2 \in \mathcal{S}$ such that

$$D^* := \gcd(x_1 - x_2, N) = \max_{x,x' \in \mathcal{S},\ x \neq x'} \gcd(x - x', N).$$

If $D^* \geqslant 2$, then $D_{\mathcal{S}} = D^*$, and our enhanced attack using x_1 and x_2 as its two oracle queries is a generic group-action algorithm with complexity $O(\sqrt{N/D_{\mathcal{S}}})$ and advantage $1 - 1/D_{\mathcal{S}}$. If $D^* = 1$, then $D_{\mathcal{S}} \in \{1, 2\}$, and hence $\sqrt{N/D_{\mathcal{S}}} = \Theta(\sqrt{N})$. Within the same asymptotic query complexity, a generic group action algorithm can solve the underlying GAIP problem using the BSGS technique.

4.2 Sparse-Poly PRFs with Restricted Input Space

In this section, we use the idea of *Removing Polynomial Terms* to further improve the efficiency and compactness of the proposed PRFs and VRFs. One may observe from the proof of Theorem 4 that not all polynomial terms play a decisive role in estimating the upper bound. Therefore, we consider the following PRFs construction.

Definition 6 ($(d_1, \ldots, d_m)$-Poly-style PRFs with Restricted Input Space) *Given a* $\mathrm{CEGA}^{\mathsf{T}}$ $(\mathbb{Z}_N, \mathcal{E}, \diamond, E_0)$*, and several positive integers* $d_1, \ldots, d_m$ *satisfying* $1 \leqslant d_1 < \cdots < d_m$*, we define that the* $(d_1, \ldots, d_m)$*-Poly-style PRFs* F^{SP} *with a restricted input space* $\mathcal{S}$ *is a keyed function*

$$F^{\mathsf{SP}}(\mathbf{k}, x) = f(x) \diamond E_0 = (a_m x^{d_m} + \cdots + a_1 x^{d_1} + a_0) \diamond E_0, \tag{11}$$

where $(\mathbb{Z}_N, \mathcal{E}, \diamond, E_0, d_1, \ldots, d_m, \mathcal{S})$ *can be seen as the public parameters,* x *is the input value in* $\mathcal{S} \subseteq \mathbb{Z}_N$ *and the coefficient vector* $\mathbf{k} = (a_0, \ldots, a_m)$ *is the key of the PRFs randomly chosen from* $\mathbb{Z}_N^{m+1}$*. In addition to this, we require that* $(d_1, \ldots, d_m, \mathcal{S})$ *is a tuple such that the map* $\theta : \mathcal{S} \to \mathbb{Z}_N^m$*,* $x \to (x^{d_1}, \ldots, x^{d_m})$ *to be injective.*

The injectivity of θ is important for the security. If this restriction is not imposed, one may readily find two distinct elements $\mathrm{x}_1, \mathrm{x}_2 \in \mathcal{S}$ such that $a_m \mathrm{x}_1^{d_m} + \cdots + a_1 \mathrm{x}_1^{d_1} + a_0 \equiv a_m \mathrm{x}_2^{d_m} + \cdots + a_1 \mathrm{x}_2^{d_1} + a_0 \pmod{N}$ holds for every $\mathbf{k} = (a_0, \ldots, a_m) \in \mathbb{Z}_N^{m+1}$. Consequently, the PRFs can be distinguished from a truly random function with advantage close to 1 by querying the oracle on inputs x_1 and x_2. A simple way to ensure that θ is injective is as follows: set the input space as $\mathcal{S} = \{0, 1, \ldots, S\}$ with $S < \sqrt[d_1]{N}$, and hence $0^{d_1} \bmod N$, $1^{d_1} \bmod N$, $\ldots$, $S^{d_1} \bmod N$ are pairwise distinct, which implies the injectivity of θ. The following theorem gives the security performance of above PRFs in the $\mathrm{GGAM}^{\mathsf{T}}$.

Theorem 5 *Consider the $(d_1, \ldots, d_m)$-Poly-style PRFs $F^{\mathsf{SP}}(\mathbf{k}, x) = (a_m x^{d_m} + \cdots + a_1 x^{d_1} + a_0) \diamond E_0$. For a subset $\mathcal{S} \subseteq \mathbb{Z}_N$ with $|\mathcal{S}| \geqslant 2$, $\mathbf{k} \in \mathbb{Z}_N^{m+1}$, and $b \in \{0,1\}$, the oracle $\mathcal{O}_{b,\mathbf{k}}^{\mathcal{S}}$ is defined in the same way as in Theorem 4, except that the PRF F^{Poly} is replaced by F^{SP}. Let $\mathcal{A}$ be a generic group action algorithm that outputs a bit b^* after making q queries to the oracle $\mathcal{O}_{b,\mathbf{k}}^{\mathcal{S}}$, and a total of q' queries to the group action oracle $\mathcal{O}^{\mathsf{GA}}$ and the twist oracle $\mathcal{O}^{\mathsf{Tw}}$. Define the advantage of $\mathcal{A}$*

$$\varepsilon_{\mathcal{A}} := 2\left(\Pr\left[b^* = b \;\middle|\; \begin{array}{c} b \xleftarrow{\$} \{0,1\},\ \mathbf{k} \xleftarrow{\$} \mathbb{Z}_N^{m+1}, \\ b^* \leftarrow \mathcal{A}^{\mathcal{O}_{b,\mathbf{k}}^{\mathcal{S}}, \mathcal{O}^{\mathsf{GA}}, \mathcal{O}^{\mathsf{Tw}}}(\mathbb{Z}_N, \sigma_{\mathcal{E}}(E_0), d_1, \ldots, d_m, \mathcal{S}) \end{array}\right] - \frac{1}{2}\right).$$

Then, for any $q, q' \in \mathbb{N}$ such that $2q + q' \leqslant \sqrt{\frac{N}{D_{\mathcal{S}}^*}} - 1$, we have

$$\varepsilon_{\mathcal{A}} = O\left(\frac{(2q+q')^2 D_{\mathcal{S}}^*}{N}\right),$$

where $D_{\mathcal{S}}^* := \max\{\gcd(2, N), \max_{\mathrm{x}, \mathrm{x}' \in \mathcal{S}, \mathrm{x} \neq \mathrm{x}'} \gcd(\mathrm{x}^{d_1} - \mathrm{x}'^{d_1}, \ldots, \mathrm{x}^{d_m} - \mathrm{x}'^{d_m}, N)\}$.

Proof. It suffices to present the computation of $D_{\mathcal{S}}^*$. The remaining argument is identical to that in the proof of Theorem 4, with the corresponding dimension changed from $d+q+1$ to $m+q+1$. Thus, we adopt the notation used therein. In the present setting, up to a sign, the polynomial $\hat{f}_i^{(b)} - \hat{f}_j^{(b)}$ is of one of the following eight forms:

$$\begin{aligned}
&(1)\ A_0 + \mathrm{x}^{d_1} A_1 + \cdots + \mathrm{x}^{d_m} A_m + c, \quad (2)\ R_i + c, \\
&(3)\ (\mathrm{x}^{d_1} - \mathrm{x}'^{d_1}) A_1 + \cdots + (\mathrm{x}^{d_m} - \mathrm{x}'^{d_m}) A_m + c \quad (\mathrm{x} \neq \mathrm{x}'), \\
&(4)\ 2A_0 + (\mathrm{x}^{d_1} + \mathrm{x}'^{d_1}) A_1 + \cdots + (\mathrm{x}^{d_m} + \mathrm{x}'^{d_m}) A_m + c, \\
&(5)\ A_0 + \mathrm{x}^{d_1} A_1 + \cdots + \mathrm{x}^{d_m} A_m \pm R_i + c, \\
&(6)\ R_i \pm R_j + c \ \ (i \neq j), \quad (7)\ 2R_i + c, \quad (8)\ c \quad (c \neq 0).
\end{aligned}$$

where $\mathrm{x}, \mathrm{x}' \in \{\mathrm{x}_i\}_{i=1}^q$, $c \in \mathbb{Z}_N$ is a constant. Based on Lemma 1, the polynomial (8) has no solutions; the polynomials (1), (2), (5) and (6) each have exactly N^{m+q} solutions; the polynomials (4) and (7) each have at most $N^{m+q} \gcd(2, N)$ solutions; the polynomial (3) has at most

$$N^{m+q} \max_{\mathrm{x}, \mathrm{x}' \in \mathcal{S}, \mathrm{x} \neq \mathrm{x}'} \gcd(\mathrm{x}^{d_1} - \mathrm{x}'^{d_1}, \ldots, \mathrm{x}^{d_m} - \mathrm{x}'^{d_m}, N)$$

solutions. Thus, we let

$$D^*_{\mathcal{S}} := \max\left\{\gcd(2,N), \max_{\mathrm{x},\mathrm{x}'\in\mathcal{S},\mathrm{x}\neq\mathrm{x}'} \gcd(\mathrm{x}^{d_1}-\mathrm{x}'^{d_1},\ldots,\mathrm{x}^{d_m}-\mathrm{x}'^{d_m},N)\right\}.$$

Applying the same argument as in the proof of Theorem 4, with $D_{\mathcal{S}}$ replaced by $D^*_{\mathcal{S}}$, yields the claimed upper bound. □

Remark: From Theorem 5, one can also observe the importance of the injectivity of θ: otherwise, one can find $x, x' \in \mathcal{S}$ such that $\mathrm{x}^{d_1}-\mathrm{x}'^{d_1} \equiv \cdots \equiv \mathrm{x}^{d_m}-\mathrm{x}'^{d_m} \equiv 0 \pmod{N}$. This implies that $D^*_{\mathcal{S}} = N$, preventing us from obtaining a meaningful security guarantee.

Choices of Polynomials and Input Space $\mathcal{S}$. We propose using the following two classes of sparse polynomials, which together cover all degrees $d \geqslant 2$.

1. $f^{(2)}(x) = a_1x^2 + a_0$, namely, 2-Poly-style PRFs $F(\mathbf{k}, x) = (a_1x^2 + a_0) \diamond E_0$.
2. $f^{(1,d)}(x) = a_2x^d + a_1x + a_0$ $(d \geqslant 2)$, namely, $(1,d)$-Poly-style PRFs $F(\mathbf{k}, x) = (a_2x^d + a_1x + a_0) \diamond E_0$.

Our choice of the above two types of polynomials is motivated by the following considerations. However, we do not rule out the security of other polynomial choices in certain scenarios, provided that the input space $\mathcal{S}$ is appropriately chosen.

– We do not consider linear polynomials ($d = 1$) because, when adversaries have quantum access to the PRF evaluation oracle, the linear coefficient can be efficiently recovered [42,21] by solving a (abelian) Hidden Subgroup Problem (HSP) instance, thereby breaking the security of the PRF. In contrast, this HSP-based attack does not directly extend to nonlinear polynomials. However, we do not claim that using quadratic or higher-degree polynomials necessarily guarantees security against adversaries with quantum query access to the PRF evaluation oracle. Moreover, it is worth noting that the choice of the input space $\mathcal{S}$ may also be an important factor affecting the attack advantage of adversaries with quantum query access. Nevertheless, in some scenarios, even if an adversary possesses quantum computing capabilities, this does not necessarily imply that the PRF evaluation oracle can be accessed quantumly, which depends on the actual security requirements. Since there is no significant efficiency gap between VRFs built from linear polynomials $f^{(1)}(x) = a_1x + a_0$ and those built from quadratic polynomials $f^{(2)}(x) = a_1x^2 + a_0$ as the base polynomials, we do not use linear polynomials in our construction.
– For polynomials of degree $d \geqslant 3$, when retaining only the terms of degrees d, 1, and 0, the intermediate terms do not improve the security bound provided by Theorem 5 in the GGAM$^{\mathsf{T}}$ model (note that the corresponding values of $D^*_{\mathcal{S}} = \max\{\gcd(2,N), \max_{\mathrm{x},\mathrm{x}'\in\mathcal{S},\mathrm{x}\neq\mathrm{x}'} \gcd(\mathrm{x}-\mathrm{x}',N)\}$ are exactly the same). Moreover, we do not rule out the potential value of omitting the linear term, i.e., using $f^{(d)}(x) = a_1x^d + a_0$. However, in this case, one needs to carefully choose the input space $\mathcal{S}$ to ensure that $\max_{\mathrm{x},\mathrm{x}'\in\mathcal{S},\mathrm{x}\neq\mathrm{x}'} \gcd(\mathrm{x}^d-\mathrm{x}'^d,N)$ $(d \geqslant 3)$ remains

sufficiently small, which involves a more complicated analysis than in the cases $d = 1$ and $d = 2$.
– One may also remove the constant term a_0 to further reduce the cost. Using the same approach as in Theorem 5, one can similarly compute the corresponding value $D^*_{\mathcal{S}}$. In this case, one should simultaneously consider the upper bound on $\max_{\mathrm{x}\in\mathcal{S}} \gcd(\mathrm{x}^{d_1}, \ldots, \mathrm{x}^{d_m}, N) = \max_{\mathrm{x}\in\mathcal{S}} \gcd(\mathrm{x}^{d_1}, N)$, $\max_{\mathrm{x},\mathrm{x}'\in\mathcal{S},\,\mathrm{x}\neq\mathrm{x}'} \gcd(\mathrm{x}^{d_1} - \mathrm{x}'^{d_1}, \ldots, \mathrm{x}^{d_m} - \mathrm{x}'^{d_m}, N)$, and $\max_{\mathrm{x},\mathrm{x}'\in\mathcal{S}} \gcd(\mathrm{x}^{d_1} + \mathrm{x}'^{d_1}, \ldots, \mathrm{x}^{d_m} + \mathrm{x}'^{d_m}, N)$. Therefore, in this case, the input space $\mathcal{S}$ should be chosen carefully. In particular, besides requiring θ to be injective, $0 \in \mathbb{Z}_N$ must also be excluded from the valid input space. However, we do not treat this case as a primary focus, since applying our expansion technique reintroduces a constant term (see Section 4.3). Nevertheless, if the corresponding input space $\mathcal{S}$ is sufficiently large so that no expansion is required, such PRFs may still offer good practical utility.

For the above two sparse polynomials $f^{(2)}$ and $f^{(1,d)}$, we show the following selection strategy for input space $\mathcal{S}$:

List all divisors $t_1, t_2, \ldots, t_{\tau(N)}$[14] of N in increasing order, and determine the target divisor $t_{\tilde{i}}$ according to the desired security level, as illustrated below:

$$t_1 = 1 < \cdots < \boxed{t_{\tilde{i}}} < t_{\tilde{i}+1} < \cdots < t_{\tau(N)} = N.$$

2. Select an efficiently encodable set $\tilde{\mathcal{S}} \subseteq \mathbb{Z}_N$ such that $\tilde{\mathcal{S}} \in \{\mathcal{S} \mid D^*_{\mathcal{S}} \leqslant t_{\tilde{i}}\}$, and for any $\mathcal{S} \in \{\mathcal{S} \mid D^*_{\mathcal{S}} \leqslant t_{\tilde{i}}\}$, we have $|\tilde{\mathcal{S}}| \geqslant |\mathcal{S}|$. Here,

$$D^*_{\mathcal{S}} = \begin{cases} \max\{\gcd(2, N), \max\limits_{\mathrm{x},\mathrm{x}'\in\mathcal{S},\mathrm{x}\neq\mathrm{x}'} \gcd(\mathrm{x} - \mathrm{x}', N)\} & \textit{for the polynomial } f^{(1,d)}, \\ \max\{\gcd(2, N), \max\limits_{\mathrm{x},\mathrm{x}'\in\mathcal{S},\mathrm{x}\neq\mathrm{x}'} \gcd(\mathrm{x}^2 - \mathrm{x}'^2, N)\} & \textit{for the polynomial } f^{(2)}. \end{cases}$$

The following theorem helps determine whether the selected set $\tilde{\mathcal{S}}$ has maximum cardinality.

Theorem 6 *Let N be a composite number with divisors $1 = t_1 < t_2 < \cdots < t_{\tau(N)} = N$, and let $1 \leqslant \tilde{i} < \tau(N)$ and $\mathcal{S} \subseteq \mathbb{Z}_N$, then:*
(1). $\max\limits_{\mathrm{x},\mathrm{x}'\in\mathcal{S},\mathrm{x}\neq\mathrm{x}'} \gcd(\mathrm{x}-\mathrm{x}', N) \leqslant t_{\tilde{i}}$ implies $|\mathcal{S}| \leqslant t_{\tilde{i}+1}$, and the equality is achieved when $\mathcal{S} = \{0, 1, \ldots, t_{\tilde{i}+1} - 1\}$.
(2). $\max\limits_{\mathrm{x},\mathrm{x}'\in\mathcal{S},\mathrm{x}\neq\mathrm{x}'} \gcd(\mathrm{x}^2 - \mathrm{x}'^2, N) \leqslant t_{\tilde{i}}$ implies $|\mathcal{S}| \leqslant \mathrm{QR}(t_{\tilde{i}+1})$, where

$$\mathrm{QR}(n) := \left|\{x^2 \bmod n \mid x \in \mathbb{Z}_n\}\right|$$

denotes the number of distinct square residues modulo n, including zero.

Proof. (1). Consider the residue classes $0 \bmod t_{\tilde{i}+1}, \ldots, (t_{\tilde{i}+1} - 1) \bmod t_{\tilde{i}+1}$. If $|\mathcal{S}| > t_{\tilde{i}+1}$, then by the pigeonhole principle, there exist $\mathrm{x}, \mathrm{x}' \in \mathcal{S}$ that belong to the same residue class. This implies $\gcd(\mathrm{x} - \mathrm{x}', N) \geqslant t_{\tilde{i}+1} > t_{\tilde{i}}$, a contradiction. Moreover, note that when $\mathcal{S} = \{0, 1, \ldots, t_{\tilde{i}+1} - 1\}$, we have $\max\limits_{\mathrm{x},\mathrm{x}'\in\mathcal{S},\mathrm{x}\neq\mathrm{x}'} \gcd(\mathrm{x} -$

[14] Here, $\tau(N)$ denotes the number of positive divisors of N.

Table 3. Candidate Sparse-Poly PRFs and Parameter Settings for CSIDH-512 and SCALLOP

Sparse-Poly PRFs	Parameter set	Input Space $\mathcal{S}$	$\lfloor \log_2 \lvert\mathcal{S}\rvert \rfloor$	$D^*_{\mathcal{S}}$	$\log_2(\sqrt{N/D^*_{\mathcal{S}}})$
$(a_2x^d + a_1x + a_0) \diamond E_0$	CSIDH-512 [7]	$\{0, 1, \ldots, t_9 - 1\}$	95	t_8	115
		$\{0, 1, \ldots, t_5 - 1\}$	20	t_4	125
	SCALLOP [32,39]	$\{0, 1, \ldots, q - 1\}$	$\lvert q\rvert$	2	$\lvert q\rvert/2$
$(a_1x^2 + a_0) \diamond E_0$	CSIDH-512 [7]	$\{0, 1, \ldots, \frac{t_9-1}{2}\}$	94	t_8	115
		$\{0, 1, \ldots, \frac{t_5-1}{2}\}$	19	t_4	125
	SCALLOP [32,39]	$\{0, 1, \ldots, \frac{q-1}{2}\}$	$\lvert q\rvert - 1$	2	$\lvert q\rvert/2$

Remark: Here, the SCALLOP parameters use the form in which the class group has order $N = 2q$, where q is a large prime. $\lvert q\rvert := \lfloor \log_2 q \rfloor$, and $\log_2(\sqrt{N/D^*_{\mathcal{S}}})$ is the security level in the GGAM$^{\mathsf{T}}$.

$\mathrm{x}', N)\} = t_{\tilde{i}}$. This implies that the inequality in the theorem is tight.
(2). Similarly, consider the residue classes $0^2 \bmod t_{\tilde{i}+1}, \ \ldots, \ (t_{\tilde{i}+1}-1)^2 \bmod t_{\tilde{i}+1}$. If $\lvert\mathcal{S}\rvert > \mathrm{QR}(t_{\tilde{i}+1})$, then there exist $\mathrm{x}, \mathrm{x}' \in \mathcal{S}$ such that $t_{\tilde{i}+1} \mid \mathrm{x}^2 - \mathrm{x}'^2$. This implies $\gcd(\mathrm{x}^2 - \mathrm{x}'^2, N) \geqslant t_{\tilde{i}+1} > t_{\tilde{i}}$, a contradiction. □

We provide a formula for computing $\mathrm{QR}(n)$ together with a proof in Appendix C. For the parameters considered in this section, it suffices to note that, for any odd prime p, $\mathrm{QR}(p) = (p+1)/2$. Below, we provide guidelines for selecting $\mathcal{S}$ under the CSIDH-512 [7], as well as the SCALLOP parameters suggested in [32,39]. For CSIDH-512, N and its factors in increasing order are

$$N = p_1p_2p_3p_4p_5 = 3 \times 37 \times 1407181 \times 5159360429529586774429358488 9 \times 3159941450468199585300827874558783220490 9$$

$$t_1 = 1, \ t_2 = 3, \ t_3 = 37, \ t_4 = 3 \times 37 = 111, \ t_5 = 1407181,$$

$$t_6 = 3 \times 1407181 = 4221543, \ t_7 = 37 \times 1407181 = 52065697,$$

$$t_8 = 3 \times 37 \times 1407181 = 156197091, \ t_9 = 5159360429529586774429358488 9, \cdots$$

For the SCALLOP parameters (where $N = 2q$ and q is a large prime), the factors in increasing order are $1, 2, q, N$. We select the restricted sets $\mathcal{S}$ for each of these two parameter settings under different security levels, as shown in Table 3.

Unlike the recent approach of Pedersen [39], we do not restrict $\mathcal{S}$ to be an additive subgroup of $(\mathbb{Z}_N, +)$. In fact, the proof of Theorem 6 shows that achieving a small $D^*_{\mathcal{S}}$ is not directly related to whether $\mathcal{S}$ forms a subgroup. We believe that the underlying reason is that, even when $\mathcal{S}$ is restricted to a subgroup, its elements are still operated on as elements of the ring $\mathbb{Z}_N$. For example, consider $(1, d)$-Poly-style PRFs under the CSIDH-512 parameters. If one chooses the subgroup of order $p_{\max} = p_5$, then this subgroup is $\{0, M, 2M, \ldots, (p_5 - 1)M\}$, where $M = p_1p_2p_3p_4$. Hence, for any distinct x, x' in the subgroup, $M \mid (\mathrm{x} - \mathrm{x}')$, and in fact $\gcd(\mathrm{x} - \mathrm{x}', N) = M$. Consequently, $D^*_{\mathcal{S}} = p_1p_2p_3p_4$, which is a 123-bit integer and is therefore far too large to yield the desired security level. In contrast, under the SCALLOP parameters with $N = 2q$, the subgroup of order q, namely $\{0, 2, \ldots, 2(q-1)\}$, forms a complete residue system modulo q. Since the only non-trivial divisor of N smaller than q is 2, this subgroup

yields $D^*_{\mathcal{S}} = 2$ for $(1, d)$-Poly-style PRFs, which has the same effect as choosing $\mathcal{S} = \{0, 1, \ldots, q-1\}$. This phenomenon, however, does not arise from the subgroup structure itself; rather, it relies on the fact that the selected set forms a complete residue system modulo q. It is also worth noting that this subgroup is unsuitable for 2-Poly-style PRFs. Since the subgroup contains q elements, whereas there are only $\mathrm{QR}(q) = (q+1)/2$ distinct square residues modulo q (including zero), there must exist distinct x, x' in the subgroup such that $\mathrm{x}^2 \equiv \mathrm{x}'^2 \pmod{q}$. Moreover, since both x and x′ are even, this implies $2q \mid (\mathrm{x}^2 - \mathrm{x}'^2)$, and hence $D^*_{\mathcal{S}} = N$. This means that, by querying x and x′ and checking whether the outputs are equal, the adversary can distinguish the PRF from a random function with advantage close to 1.

4.3 The Expansion Technique

As can be seen from the parameter selection part (Table 3, Section 4.2), the size of the input space may become small when we target a high security level. Therefore, in this section, we introduce our expansion technique, which expands the input space $\mathcal{S}$ of the original Sparse-Poly PRFs by a factor l, resulting in the extended input space $\mathcal{S}^l$. Moreover, we provide a reduction showing that breaking the pseudorandomness of the expanded PRFs is at least as hard as breaking that of the original Sparse-Poly PRFs under the GA-DDH assumption.

Specifically, for a given $\mathrm{CEGA}^{\mathsf{T}}$ $(\mathbb{Z}_N, \mathcal{E}, \diamond, E_0)$, a subset $\mathcal{S} \subseteq \mathbb{Z}_N$, a space $\mathcal{K}'$, and a set of keyed functions $\mathcal{F} = \{f_{\mathbf{k}'}\}_{\mathbf{k}' \in \mathcal{K}'}$, where every function $f_{\mathbf{k}'} \in \mathcal{F}$ is an efficiently computable function from $\mathcal{S}$ to $\mathbb{Z}_N$ and there is an efficient algorithm that can uniformly sample a key $\mathbf{k}' \in \mathcal{K}'$. Consider the PRFs $F : \mathcal{K} \times \mathcal{S} \to \mathcal{E}$ with the generic structure as follows:

$$F(\mathbf{k}, x) = (c + f_{\mathbf{k}'}(x)) \diamond E_0 \tag{12}$$

where $\mathcal{K} = \mathbb{Z}_N \times \mathcal{K}'$ is the key space of this function, $\mathbf{k} = (c, \mathbf{k}') \xleftarrow{\$} \mathcal{K}$ is the evaluation key, and $\mathcal{S}$ is the valid input space[15].

Our expansion technique is applicable to any PRFs of the above forms in (12). Let $l \geqslant 2$ be a positive integer denoting the expansion factor. We define the expanded PRFs as $F^{\mathsf{ex}} : \mathcal{K}^{\mathsf{ex}} \times \mathcal{S}^l \to \mathcal{E}$

$$F^{\mathsf{ex}}(\mathbf{k}, x) = \big(c + f_{\mathbf{k}'_1}(x_1) + \cdots + f_{\mathbf{k}'_l}(x_l)\big) \diamond E_0. \tag{13}$$

Here, the expanded key space is $\mathcal{K}^{\mathsf{ex}} = \mathbb{Z}_N \times (\mathcal{K}')^l$, and the evaluation key $\mathbf{k} = (c, \mathbf{k}'_1, \ldots, \mathbf{k}'_l) \xleftarrow{\$} \mathcal{K}^{\mathsf{ex}}$. The expanded input space is $\mathcal{S}^l$, and the input $x = x_1 \| \cdots \| x_l \in \mathcal{S}^l$ with $x_i \in \mathcal{S}$ for all $i \in [l]$. The output space $\mathcal{E}$ remains unchanged.

[15] *Remark:* One may consider PRFs (12) with the constant term removed, namely, $F(\mathbf{k}', x) = f_{\mathbf{k}'}(x) \diamond E_0$, where $\mathbf{k}' \in \mathcal{K}' = \mathcal{K}$. This form is also compatible with our expansion technique. However, somewhat counterintuitively, the resulting expanded PRFs are still PRFs (13) (containing a constant term). This can be observed from the proof of Theorem 10.

To prove the security of our expanded PRF construction, we first introduce an interesting hardness assumption and show that it is implied by the GA-DDH assumption.

Definition 7 (Group-Action separable-randomness Decisional Diffie-Hellman (GA-srDDH)) *Let $(\mathbb{Z}_N, \mathcal{E}, \diamond, E_0)$ be a $\mathrm{CEGA}^{\mathsf{T}}$, and let $\mathcal{X}_1$ and $\mathcal{X}_2$ be two input spaces. For $b \in \{0,1\}$, define the oracle $\mathcal{O}_b^{\mathsf{SR}} : \mathcal{X}_1 \times \mathcal{X}_2 \to \mathcal{E}$ as follows:*

$$\mathcal{O}_b^{\mathsf{SR}}(x,y) = \begin{cases} t(x,y) \diamond E_0, & \text{if } b = 0, \\ \big(r(x) + s(y)\big) \diamond E_0, & \text{if } b = 1, \end{cases}$$

where $r \xleftarrow{\$} \mathrm{Func}(\mathcal{X}_1, \mathbb{Z}_N)$, $s \xleftarrow{\$} \mathrm{Func}(\mathcal{X}_2, \mathbb{Z}_N)$, and $t \xleftarrow{\$} \mathrm{Func}(\mathcal{X}_1 \times \mathcal{X}_2, \mathbb{Z}_N)$ are mutually independent truly random functions. We say that the GA-srDDH *assumption holds if, for every PPT adversary $\mathcal{A}$, there exists a negligible function* $\mathrm{negl}(\lambda)$ *such that*

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{GA\text{-}srDDH}}(\lambda) := 2\left(\Pr[b^* = b \mid b \xleftarrow{\$} \{0,1\},\ b^* \leftarrow \mathcal{A}^{\mathcal{O}_b^{\mathsf{SR}}}(1^\lambda)] - \frac{1}{2}\right) \leqslant \mathrm{negl}(\lambda).$$

Remark: It is clear that one can easily distinguish a truly random function $t(x,y)$ from the separable function $t'(x,y) := r(x) + s(y)$ by querying the four points (x_1,y_1), (x_1,y_2), (x_2,y_1), and (x_2,y_2), and checking whether $\mathcal{O}(x_1,y_1) + \mathcal{O}(x_2,y_2) = \mathcal{O}(x_1,y_2) + \mathcal{O}(x_2,y_1)$, where the oracle $\mathcal{O}$ is either t or t'. However, we prove that, in the group-action setting, the functions $t(x,y) \diamond E_0$ and $t'(x,y) \diamond E_0$ are computationally indistinguishable under the GA-DDH assumption. To prove this, we introduce the following hardness assumption.

Definition 8 Group-Action (m,n)-matrix Decisional Diffie-Hellman (GA-(m,n)-matrix-DDH) *Let $(\mathbb{Z}_N, \mathcal{E}, \diamond, E_0)$ be a $\mathrm{CEGA}^{\mathsf{T}}$, and fix two integers $m, n \geqslant 2$. Let*

$$\mathbf{A}_0 = \begin{pmatrix} r_{1,1} \diamond E_0 & \cdots & r_{1,n} \diamond E_0 \\ \vdots & \ddots & \vdots \\ r_{m,1} \diamond E_0 & \cdots & r_{m,n} \diamond E_0 \end{pmatrix}, \qquad \mathbf{A}_1 = \begin{pmatrix} (a_1 + b_1) \diamond E_0 & \cdots & (a_1 + b_n) \diamond E_0 \\ \vdots & \ddots & \vdots \\ (a_m + b_1) \diamond E_0 & \cdots & (a_m + b_n) \diamond E_0 \end{pmatrix},$$

where $a_1, \ldots, a_m, b_1, \ldots, b_n, r_{1,1}, \ldots, r_{m,n}$ are sampled independently and uniformly at random from $\mathbb{Z}_N$. We say that the GA-(m,n)-matrix-DDH *assumption holds if, for every PPT adversary $\mathcal{A}$, there exists a negligible function* $\mathrm{negl}(\lambda)$ *such that*

$$\mathbf{Adv}_{\mathcal{A}}^{(m,n)\text{-}\mathsf{mDDH}}(\lambda) := 2\left(\Pr[b^* = b \mid b \xleftarrow{\$} \{0,1\},\ b^* \leftarrow \mathcal{A}(1^\lambda, \mathbf{A}_b)] - \frac{1}{2}\right) \leqslant \mathrm{negl}(\lambda).$$

Theorem 7 *Let $\mathcal{A}$ be a PPT algorithm that wins the* GA-srDDH *game with advantage ε by making q oracle queries. Then, there exists a PPT algorithm $\mathcal{B}$ such that $\mathcal{B}^{\mathcal{A}}$ wins the* GA-(q,q)-matrix-DDH *game with the same advantage ε.*

Proof. Given a GA-(q,q)-matrix-DDH instance $\mathbf{A}_b = (z_{i,j} \diamond E_0)_{i,j\in[q]}$, where either $z_{i,j} = r_{i,j}$ for all $i,j \in [q]$, or $z_{i,j} = a_i + b_j$ for all $i,j \in [q]$, $\mathcal{B}$ answers $\mathcal{A}$'s oracle queries according to the following rules:

– Before answering $\mathcal{A}$'s queries, $\mathcal{B}$ initializes two empty lists $\mathcal{L}_{\mathsf{left}}$ and $\mathcal{L}_{\mathsf{right}}$.
– When $\mathcal{A}$ makes a query (x,y), $\mathcal{B}$ first checks whether there exists an index $\mathsf{ind} \in [k]$ such that $x_{\mathsf{ind}} = x$, assuming that $\mathcal{L}_{\mathsf{left}}$ currently contains k ($k \geqslant 0$) distinct values. If such an index exists, $\mathcal{B}$ sets $i \leftarrow \mathsf{ind}$. Otherwise, $\mathcal{B}$ sets $x_{k+1} \leftarrow x$, updates $\mathcal{L}_{\mathsf{left}} \leftarrow \mathcal{L}_{\mathsf{left}} \cup \{x_{k+1}\}$, and sets $i \leftarrow k+1$. Similarly, $\mathcal{B}$ then checks whether there exists an index $\mathsf{ind} \in [l]$ such that $y_{\mathsf{ind}} = y$, assuming that $\mathcal{L}_{\mathsf{right}}$ currently contains l ($l \geqslant 0$) distinct values. If such an index exists, $\mathcal{B}$ sets $j \leftarrow \mathsf{ind}$. Otherwise, $\mathcal{B}$ sets $y_{l+1} \leftarrow y$, updates $\mathcal{L}_{\mathsf{right}} \leftarrow \mathcal{L}_{\mathsf{right}} \cup \{y_{l+1}\}$, and sets $j \leftarrow l+1$. Finally, $\mathcal{B}$ returns $z_{i,j} \diamond E_0$ to $\mathcal{A}$.
– After $\mathcal{A}$ completes its q queries, it outputs a guess b'. Then, $\mathcal{B}$ simply outputs the same guess $b^* = b'$.

It is easy to see that $i,j \leqslant q$ for every query. In other words, all q queries made by $\mathcal{A}$ can be embedded into the $q \times q$ matrix of the GA-(q,q)-matrix-DDH instance $\mathbf{A}_b$, yielding a perfect simulation for GA-srDDH game. This implies that the above algorithm $\mathcal{B}^{\mathcal{A}}$ solves the GA-(q,q)-matrix-DDH problem with the same advantage ε as $\mathcal{A}$ in solving the GA-srDDH problem. □

Remark: It is easy to see that we can also use an algorithm $\mathcal{A}$ that solves the GA-(m,n)-matrix-DDH problem to solve the GA-srDDH problem by querying the GA-srDDH oracle $\mathcal{O}_b^{\mathsf{SR}}$ on all pairs $\{(i,j)\}_{i\in[m],j\in[n]}$.

Theorem 8 *Let $\mathcal{A}$ be a PPT algorithm that wins the* GA-(m,n)-matrix-DDH *game with advantage ε. Then, there exists a PPT algorithm $\mathcal{B}$ such that $\mathcal{B}^{\mathcal{A}}$ solves the* GA-DDH *problem with advantage $\frac{1}{(m-1)(n-1)}\varepsilon$.*

Proof. First, we recall the Multi-Challenge Group Action Decisional Diffie-Hellman (GA-mcDDH) assumption introduced by Lai in his work on VRFs [28]. In this proof, it suffices for us to adopt the following static variant of Lai's definition. Interested readers may refer to the original work for a comparison.

Let $(\mathbb{Z}_N, \mathcal{E}, \diamond, E_0)$ be a CEGA$^{\mathsf{T}}$, and let

$$\mathbf{v}_0 = (a_1 \diamond E_0, \ldots, a_q \diamond E_0, r_1 \diamond E_0, \ldots, r_q \diamond E_0),$$
$$\mathbf{v}_1 = (a_1 \diamond E_0, \ldots, a_q \diamond E_0, (a_1 + r_0) \diamond E_0, \ldots, (a_q + r_0) \diamond E_0),$$

where $q \geqslant 2$ is an integer, and $a_1, \ldots, a_q, r_0, r_1, \ldots, r_q$ are sampled independently and uniformly at random from $\mathbb{Z}_N$. We say that the GA-q-mcDDH assumption holds if, for every PPT adversary $\mathcal{A}$, there exists a negligible function $\mathrm{negl}(\lambda)$ such that

$$\mathbf{Adv}_{\mathcal{A}}^{q\text{-}\mathsf{mcDDH}}(\lambda) := 2\left(\Pr[b^* = b \mid b \xleftarrow{\$} \{0,1\},\ b^* \leftarrow \mathcal{A}(1^\lambda, \mathbf{v}_b)] - \frac{1}{2}\right) \leqslant \mathrm{negl}(\lambda).$$

Assume that $\mathcal{A}$ is an algorithm that achieves advantage ε in the GA-(m,n)-matrix-DDH game. We construct an efficient algorithm $\mathcal{B}_1$ such that $\mathcal{B}_1^{\mathcal{A}}$ solves the GA-m-mcDDH problem with advantage $\frac{1}{n-1}\varepsilon$.

$\mathcal{B}_1$ is given a random GA-m-mcDDH problem instance $\mathbf{v}_b = (a_1 \diamond E_0, \ldots, a_m \diamond E_0, z_{1,b} \diamond E_0, \ldots, z_{m,b} \diamond E_0)$, where $(z_{1,0}, \ldots, z_{m,0}) = (r_1, \ldots, r_m)$, $(z_{1,1}, \ldots, z_{m,1}) = (a_1 + r_0, \ldots, a_m + r_0)$, and $b \xleftarrow{\$} \{0,1\}$. $\mathcal{B}_1$ randomly selects $J \xleftarrow{\$} [2,n]$, independently sample $s_1, \ldots, s_{J-1} \xleftarrow{\$} \mathbb{Z}_N$ and $s_{i,j} \xleftarrow{\$} \mathbb{Z}_N$ for every $(i,j) \in [1,m] \times [J+1,n]$. Then, it computes the following matrix $\mathbf{A}^*$

$$\begin{pmatrix} (a_1+s_1)\diamond E_0 & \cdots & (a_1+s_{J-1})\diamond E_0 & z_{1,b}\diamond E_0 & s_{1,J+1}\diamond E_0 & \cdots & s_{1,n}\diamond E_0 \\ \vdots & & \vdots & \vdots & \vdots & \ddots & \vdots \\ (a_m+s_1)\diamond E_0 & \cdots & (a_m+s_{J-1})\diamond E_0 & z_{m,b}\diamond E_0 & s_{m,J+1}\diamond E_0 & \cdots & s_{m,n}\diamond E_0 \end{pmatrix}$$

and send $\mathbf{A}^*$ to $\mathcal{A}$ as the challenge matrix. Finally, $\mathcal{A}$ outputs a guess b', and $\mathcal{B}_1$ outputs the same guess $b^* = b'$.

Observe that $\mathcal{B}_1$ simulates the J-th column using $(z_{1,b} \diamond E_0, \ldots, z_{m,b} \diamond E_0)$. Moreover, from $\mathcal{A}$'s perspective, the simulated matrices $\mathbf{A}^*$ for $J = j,\ b = 1$ and $J = j+1,\ b = 0$ are perfectly indistinguishable, and thus for every $j \in [2, n-1]$

$$\Pr[0 \leftarrow \mathcal{A} \mid J = j, b = 1] = \Pr[0 \leftarrow \mathcal{A} \mid J = j+1, b = 0]$$

Therefore, we have

$$\begin{aligned}
\Pr[\mathcal{B}_1 \text{ wins}] &= \frac{1}{n-1}\sum_{j=2}^{n} \Pr[\mathcal{B}_1 \text{ wins} \mid J = j] \\
&= \frac{1}{2(n-1)}\sum_{j=2}^{n} (\Pr[b^* = 0 \mid J = j, b = 0] + \Pr[b^* = 1 \mid J = j, b = 1]) \\
&= \frac{1}{2} + \frac{1}{2(n-1)}\sum_{j=2}^{n} (\Pr[0 \leftarrow \mathcal{A} \mid J = j, b = 0] - \Pr[0 \leftarrow \mathcal{A} \mid J = j, b = 1]) \\
&= \frac{1}{2} + \frac{1}{2(n-1)} (\Pr[0 \leftarrow \mathcal{A} \mid J = 2, b = 0] - \Pr[0 \leftarrow \mathcal{A} \mid J = n, b = 1])
\end{aligned}$$

Note that $\mathbf{A}^*$ perfectly simulates $\mathbf{A}_0$ in the case $J = 2,\ b = 0$, whereas it perfectly simulates $\mathbf{A}_1$ in the case $J = n,\ b = 1$. Therefore, the advantage of $\mathcal{B}_1^{\mathcal{A}}$ in solving the GA-m-mcDDH problem is

$$\varepsilon' = 2\left(\Pr[\mathcal{B}_1 \text{ wins}] - \frac{1}{2}\right) = \frac{1}{n-1}\varepsilon.$$

Given such $\mathcal{B}_1^{\mathcal{A}}$ solving the GA-m-mcDDH problem with advantage ε', we construct an efficient algorithm $\mathcal{B}_2$ such that $\mathcal{B}_2^{\mathcal{B}_1^{\mathcal{A}}}$ solves the GA-DDH problem with advantage $\frac{1}{m-1}\varepsilon'$.

$\mathcal{B}_2$ is given a random GA-DDH problem instance $(E_0, u \diamond E_0, v \diamond E_0, z_\delta \diamond E_0)$ $(\delta \xleftarrow{\$} \{0,1\})$, where $z_0 = r$, and $z_1 = u + v$. $\mathcal{B}_2$ randomly selects $J \xleftarrow{\$} [2,m]$,

independently sample $t_1, \ldots, t_{J-1} \xleftarrow{\$} \mathbb{Z}_N$, and $w_{i,j} \xleftarrow{\$} \mathbb{Z}_N$ for every $(i,j) \in [1,2] \times [J+1, m]$. Then, it computes the following tuple $\mathbf{v}^*$

$$\Big(t_1 \diamond E_0, \quad \ldots, \quad t_{J-1} \diamond E_0, \quad v \diamond E_0, \; w_{1,J+1} \diamond E_0, \quad \ldots, \quad w_{1,m} \diamond E_0,$$
$$(t_1 + u) \diamond E_0, \quad \ldots, \quad (t_{J-1} + u) \diamond E_0, \quad z_\delta \diamond E_0, \; w_{2,J+1} \diamond E_0, \quad \ldots, \quad w_{2,m} \diamond E_0 \Big),$$

and send $\mathbf{v}^*$ to $\mathcal{B}_1^{\mathcal{A}}$ as the challenge tuple. Finally, $\mathcal{B}_1^{\mathcal{A}}$ outputs a guess δ', and $\mathcal{B}_2$ outputs the same guess $\delta^* = \delta'$.

Observe that $\mathcal{B}_2$ simulates the J-th and $(J+m)$-th components using $v \diamond E_0$ and $z_\delta \diamond E_0$, respectively. Moreover, from $\mathcal{B}_1^{\mathcal{A}}$'s perspective, the simulated tuple $\mathbf{v}^*$ for $J = j,\ \delta = 1$ and $J = j+1,\ \delta = 0$ are perfectly indistinguishable. Therefore, the probability analysis is similar to that of the previous reduction. Omitting the details, we obtain the advantage of $\mathcal{B}_2^{\mathcal{B}_1^{\mathcal{A}}}$ in solving the GA-DDH problem is

$$\varepsilon'' = \frac{1}{m-1}\varepsilon' = \frac{1}{(m-1)(n-1)}\varepsilon,$$

which completes the proof of the theorem. □

By Theorems 7 and 8, we obtain the following Theorem 9.

Theorem 9 *Assume that there exists a PPT algorithm $\mathcal{A}$ that solves the* GA-srDDH *problem by making $q \geqslant 2$ oracle queries*[16] *with advantage ε. Then, there exists a PPT algorithm $\mathcal{B}$ such that $\mathcal{B}^{\mathcal{A}}$ solves the* GA-DDH *with advantage $\frac{1}{(q-1)^2}\varepsilon$.*

We are now ready to prove the security of our expanded PRFs (13), as stated in the following theorem.

Theorem 10 *If there exists a PPT algorithm $\mathcal{A}$ that breaks the pseudorandomness of the expanded PRFs (13) (with the input space $\mathcal{S}^l$) with advantage $\varepsilon_{\mathcal{A}}$, then, under the* GA-DDH *assumption, there exists a PPT algorithm $\mathcal{B}$ such that $\mathcal{B}^{\mathcal{A}}$ that breaks the pseudorandomness of the PRF (12) (with input space $\mathcal{S}$) with advantage $\varepsilon_{\mathcal{B}} \geqslant \frac{1}{l}\varepsilon_{\mathcal{A}} - \mathrm{negl}(\lambda)$. Therefore, when $\varepsilon_{\mathcal{A}}$ is non-negligible, so is $\varepsilon_{\mathcal{B}}$.*

Theorem 10 implies that if the GA-DDH assumption holds and the base PRFs (12) are pseudorandom, then the expanded PRFs (13) are also pseudorandom. The proof of Theorem 10 is divided into two steps. In the first step, we formally define the problem $\Lambda(\mathcal{F}, \mathcal{S})$ and prove that (Theorem 11), under the GA-DDH assumption, solving this problem is at least as hard as winning the pseudorandomness experiment of the base PRFs (12). In the second step, we prove that (Theorem 12) breaking the pseudorandomness of the expanded PRFs (13) is at least as hard as solving the problem $\Lambda(\mathcal{F}, \mathcal{S})$.

[16] Note that when $q = 1$, the advantage of any algorithm in distinguishing GA-srDDH is zero.

Definition 9 (Problem $\Lambda(\mathcal{F},\mathcal{S})$) *$\mathcal{A}$ is given a* $\mathrm{CEGA}^{\mathsf{T}}$ $(\mathbb{Z}_N,\mathcal{E},\diamond,E_0)$*, a subset $\mathcal{S}\subseteq\mathbb{Z}_N$, a key space $\mathcal{K}'$, and a family of functions $\mathcal{F}=\{f_{\mathbf{k}'}\}_{\mathbf{k}'\in\mathcal{K}'}$, defined in the same way as we define $\mathcal{F}$ for PRFs (12). For $\mathcal{S}\subseteq\mathbb{Z}_N$, $\mathbf{k}'\in\mathcal{K}'$, and $b\in\{0,1\}$, define the oracles $\mathcal{O}_{b,\mathbf{k}'}^{\Lambda(\mathcal{F},\mathcal{S})}$ as shown in the following figure. The oracle $\mathcal{O}_{b,\mathbf{k}'}^{\Lambda(\mathcal{F},\mathcal{S})}$ internally maintains a list $\mathcal{L}_b$, which is initialized to the empty set.*

Input (Ind,x) — Oracle $\mathcal{O}_{0,\mathbf{k}'}^{\Lambda(\mathcal{F},\mathcal{S})}(\cdot,\cdot)$	Input (Ind,x) — Oracle $\mathcal{O}_{1,\mathbf{k}'}^{\Lambda(\mathcal{F},\mathcal{S})}(\cdot,\cdot)$
If $x\notin\mathcal{S}$ then return $\perp$	If $x\notin\mathcal{S}$ then return $\perp$
Else check if there exists $(\mathsf{Ind}',x',r_{\mathsf{Ind}',x'})$ $\in\mathcal{L}_0$ such that $(\mathsf{Ind},x)=(\mathsf{Ind}',x')$	Else check if there exists $(\mathsf{Ind}',c_{\mathsf{Ind}'})\in\mathcal{L}_1$ such that $\mathsf{Ind}=\mathsf{Ind}'$
If so return $Y\leftarrow r_{\mathsf{Ind}',x'}\diamond E_0$	If so return $Y=(c_{\mathsf{Ind}'}+f_{\mathbf{k}'}(x))\diamond E_0$
Otherwise sample $r_{\mathsf{Ind},x}\xleftarrow{\$}\mathbb{Z}_N$ and add pair $(\mathsf{Ind},x,r_{\mathsf{Ind},x})$ to the list $\mathcal{L}_0$	Otherwise sample $c_{\mathsf{Ind}}\xleftarrow{\$}\mathbb{Z}_N$ and add the pair $(\mathsf{Ind},c_{\mathsf{Ind}})$ to the list $\mathcal{L}_1$
Return $Y\leftarrow r_{\mathsf{Ind},x}\diamond E_0$	Return $Y=(c_{\mathsf{Ind}}+f_{\mathbf{k}'}(x))\diamond E_0$

$\mathcal{A}$ may query $\mathcal{O}_{b,\mathbf{k}'}^{\Lambda(\mathcal{F},\mathcal{S})}$ before outputting a bit b^* as its guess for b. The advantage of $\mathcal{A}$ is defined as

$$\varepsilon=2\left(\Pr\left[b^*=b\;\middle|\;\begin{array}{c}b\xleftarrow{\$}\{0,1\},\ \mathbf{k}'\xleftarrow{\$}\mathcal{K}',\\ b^*\leftarrow\mathcal{A}^{\mathcal{O}_{b,\mathbf{k}'}^{\Lambda(\mathcal{F},\mathcal{S})}}((\mathbb{Z}_N,\mathcal{E},\diamond,E_0),\mathcal{F},\mathcal{S},\mathcal{K}')\end{array}\right]-\frac{1}{2}\right)$$

where the probability is taken over the randomness of $\mathcal{A}$ and the experiment.

Theorem 11 *If there exists a PPT algorithm $\mathcal{A}$ that solves the problem $\Lambda(\mathcal{F},\mathcal{S})$ with advantage $\varepsilon_{\mathcal{A}}$, then, under the* GA-DDH *assumption, there exists a PPT algorithm $\mathcal{B}$ such that $\mathcal{B}^{\mathcal{A}}$ that wins the pseudorandomness experiment (Definition 4) of the PRFs (12) over the input space $\mathcal{S}$ with advantage $\varepsilon_{\mathcal{B}}\geqslant\varepsilon_{\mathcal{A}}-\mathrm{negl}(\lambda)$.*

Proof. $\mathcal{B}$ is given the $\mathrm{CEGA}^{\mathsf{T}}$ $(\mathbb{Z}_N,\mathcal{E},\diamond,E_0)$ and access to the evaluation oracle $\mathcal{O}_{b,\mathbf{k}}^{\mathsf{PR}}$ with respect to the PRFs (12) over the input space $\mathcal{S}$. It then initializes a list $\mathcal{L}\leftarrow\varnothing$ and interacts with $\mathcal{A}$ as follows. Whenever $\mathcal{A}$ makes a query (Ind,x) to $\mathcal{B}$, $\mathcal{B}$ queries the oracle $\mathcal{O}_{b,\mathbf{k}}^{\mathsf{PR}}$ on input x and receives an output Y, which equals $(c+f_{\mathbf{k}'}(x))\diamond E_0$ if $b=1$, and $s(x)\diamond E_0$ if $b=0$. Here, $\mathbf{k}=(c,\mathbf{k}')$ and $s\xleftarrow{\$}\mathrm{Func}(\mathcal{S},\mathbb{Z}_N)$ is a truly random function. Then, $\mathcal{B}$ checks whether there exists a tuple $(\mathsf{Ind}',r_{\mathsf{Ind}'})\in\mathcal{L}$ such that $\mathsf{Ind}=\mathsf{Ind}'$. If such a tuple exists, $\mathcal{B}$ responds to $\mathcal{A}$ with $r_{\mathsf{Ind}'}\diamond Y$. Otherwise, $\mathcal{B}$ samples $r_{\mathsf{Ind}}\xleftarrow{\$}\mathbb{Z}_N$, and responds to $\mathcal{A}$ with $r_{\mathsf{Ind}}\diamond Y$. After answering the query, $\mathcal{B}$ adds the pair $(\mathsf{Ind},r_{\mathsf{Ind}})$ to the list $\mathcal{L}$. After $\mathcal{A}$ completes its queries, it sends a guess bit b' to $\mathcal{B}$. Then $\mathcal{B}$ outputs $b^*=b'$. Note that the list $\mathcal{L}$ maintained by $\mathcal{B}$ perfectly simulates polynomially many input-output pairs of a truly random function from the index space to $\mathbb{Z}_N$, which we denote by r. It is easy to see that when $b=1$, $r(\mathsf{ind})\diamond Y=(r(\mathsf{ind})+c+f_{\mathbf{k}'}(x))\diamond E_0$, and thus $\mathcal{B}$ provides $\mathcal{A}$ with a perfect simulation of the real experiment of $\Lambda(\mathcal{F},\mathcal{S})$. When $b=0$, $r(\mathsf{ind})\diamond Y=(r(\mathsf{ind})+s(x))\diamond E_0$, thus the simulation is computationally indistinguishable from the real experiment under the GA-srDDH assumption, and hence, by Theorem 9, also under the

GA-DDH assumption. Therefore, $\mathcal{B}^{\mathcal{A}}$ breaks the pseudorandomness experiment with advantage $\varepsilon_{\mathcal{B}} \geqslant \varepsilon_{\mathcal{A}} - \mathrm{negl}(\lambda)$. □

Theorem 12 *If there exists a PPT algorithm $\mathcal{A}$ that breaks the pseudorandomness (Definition 4) of the expanded PRFs (13) over the input space $\mathcal{S}^l$ with advantage $\varepsilon_{\mathcal{A}}$, then there exists a PPT algorithm $\mathcal{B}$ such that $\mathcal{B}^{\mathcal{A}}$ solves the problem $\Lambda(\mathcal{F}, \mathcal{S})$ with advantage $\varepsilon_{\mathcal{B}} = \frac{1}{l}\varepsilon_{\mathcal{A}}$.*

Proof. Assume that there exists a PPT adversary $\mathcal{A}$ as described in the theorem. Given the CEGA$^{\mathsf{T}}$ $(\mathbb{Z}_N, \mathcal{E}, \diamond, E_0)$ and the oracle $\mathcal{O}_{b,\mathbf{k}'}^{\Lambda(\mathcal{F},\mathcal{S})}$, $\mathcal{B}$ interacts with $\mathcal{A}$ and outputs a guess b^* for problem $\Lambda(\mathcal{F}, \mathcal{S})$ in the following way.
1. $\mathcal{B}$ randomly chooses $J \xleftarrow{\$} [l]$, and $(\mathbf{k}'_{J+1}, \ldots, \mathbf{k}'_l) \xleftarrow{\$} \mathcal{K}'^{\,l-J}$.
2. Whenever $\mathcal{A}$ queries $\mathcal{B}$ on input $x = x_1 \| \cdots \| x_l$, $\mathcal{B}$ responds as follows:
– If $J \in [2, l-1]$, $\mathcal{B}$ queries $\mathcal{O}_{b,\mathbf{k}'}^{\Lambda(\mathcal{F},\mathcal{S})}$ on input $(x_1 \| \ldots \| x_{J-1}, x_J)$ and responds to $\mathcal{A}$ with $(f_{\mathbf{k}'_{J+1}}(x_{J+1}) + \cdots + f_{\mathbf{k}'_l}(x_l)) \diamond \mathcal{O}_{b,\mathbf{k}'}^{\Lambda(\mathcal{F},\mathcal{S})}(x_1 \| \ldots \| x_{J-1}, x_J)$.
– If $J = 1$, $\mathcal{B}$ queries $\mathcal{O}_{b,\mathbf{k}'}^{\Lambda(\mathcal{F},\mathcal{S})}$ on input $(0, x_1)$ and responds to $\mathcal{A}$ with $(f_{\mathbf{k}'_2}(x_2) + \cdots + f_{\mathbf{k}'_l}(x_l)) \diamond \mathcal{O}_{b,\mathbf{k}'}^{\Lambda(\mathcal{F},\mathcal{S})}(0, x_1)$.
– If $J = l$, $\mathcal{B}$ queries $\mathcal{O}_{b,\mathbf{k}'}^{\Lambda(\mathcal{F},\mathcal{S})}$ on input $(x_1 \| \ldots \| x_{l-1}, x_l)$, and responds to $\mathcal{A}$ with $\mathcal{O}_{b,\mathbf{k}'}^{\Lambda(\mathcal{F},\mathcal{S})}(x_1 \| \ldots \| x_{l-1}, x_l)$.

3. After making a polynomial number of queries, $\mathcal{A}$ outputs a guess b'. Algorithm $\mathcal{B}$ then sets $b^* = b'$ and outputs b^* as its guess for b.

Now we calculate the advantage of $\mathcal{B}^{\mathcal{A}}$ in solving the problem $\Lambda(\mathcal{F}, \mathcal{S})$. It follows that

$$\begin{aligned}
&\Pr[\mathcal{B} \text{ wins}] = \frac{1}{l}\sum_{j=1}^{l} \Pr[\mathcal{B} \text{ wins} \mid J = j] \\
&= \frac{1}{2l}\sum_{j=1}^{l}(\Pr[b' = b^* = 1 \mid J = j, b = 1] + \Pr[b' = b^* = 0 \mid J = j, b = 0]) \qquad (14) \\
&= \frac{1}{2} + \frac{1}{2l}\sum_{j=1}^{l}(\Pr[1 \leftarrow \mathcal{A} \mid J = j, b = 1] - \Pr[1 \leftarrow \mathcal{A} \mid J = j, b = 0]).
\end{aligned}$$

Note that in the case where $J = j + 1$ and $b = 1$, $\mathcal{A}$ receives

$$\begin{aligned}
&(f_{\mathbf{k}'_{j+2}}(x_{j+2}) + \cdots + f_{\mathbf{k}'_l}(x_l)) \diamond \mathcal{O}_{1,\mathbf{k}'}^{\Lambda(\mathcal{F},\mathcal{S})}(x_1 \| \ldots \| x_j, x_{j+1}) \\
=&(c_{x_1\|\cdots\|x_j} + f_{\mathbf{k}'}(x_{j+1}) + f_{\mathbf{k}'_{j+2}}(x_{j+2}) + \cdots + f_{\mathbf{k}'_l}(x_l)) \diamond E_0.
\end{aligned}$$

In the case where $J = j$ and $b = 0$, $\mathcal{A}$ receives

$$\begin{aligned}
&(f_{\mathbf{k}'_{j+1}}(x_{j+1}) + \cdots + f_{\mathbf{k}'_l}(x_l)) \diamond \mathcal{O}_{0,\mathbf{k}'}^{\Lambda(\mathcal{F},\mathcal{S})}(x_1 \| \ldots \| x_{j-1}, x_j) \\
=&(r_{x_1\|\ldots\|x_{j-1},x_j} + f_{\mathbf{k}'_{j+1}}(x_{j+1}) + f_{\mathbf{k}'_{j+2}}(x_{j+2}) + \cdots + f_{\mathbf{k}'_l}(x_l)) \diamond E_0.
\end{aligned}$$

These two cases are perfectly indistinguishable to $\mathcal{A}$. Hence, for $j \in [l-1]$,

$$\Pr[1 \leftarrow \mathcal{A} \mid J = j + 1, b = 1] = \Pr[1 \leftarrow \mathcal{A} \mid J = j, b = 0]. \qquad (15)$$

From Equations (14) and (15), it follows that

$$\Pr[\mathcal{B} \text{ wins}] = \frac{1}{2} + \frac{1}{2l}\left(\Pr[1 \leftarrow \mathcal{A} \mid J = 1, b = 1] - \Pr[1 \leftarrow \mathcal{A} \mid J = l, b = 0]\right).$$

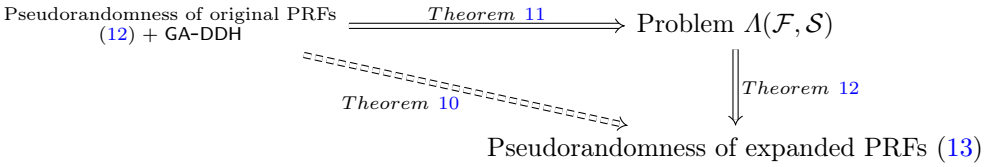
Note that in the cases where $(J = 1, b = 1)$ and $(J = l, b = 0)$, $\mathcal{B}$ perfectly simulates the evaluation oracles $\mathcal{O}^{\mathsf{PR}}_{1,\mathbf{k}}$ and $\mathcal{O}^{\mathsf{PR}}_{0,\mathbf{k}}$, respectively, for the expanded PRF (13). Thus, we have

$$\varepsilon_{\mathcal{A}} = \Pr[1 \leftarrow \mathcal{A} \mid J = 1, b = 1] - \Pr[1 \leftarrow \mathcal{A} \mid J = l, b = 0], \text{ and thus}$$

$$\varepsilon_{\mathcal{B}} = 2\big(\Pr[\mathcal{B} \text{ wins}] - \frac{1}{2}\big) = 2\big(\frac{1}{2} + \frac{1}{2l}\varepsilon_{\mathcal{A}} - \frac{1}{2}\big) = \frac{1}{l}\varepsilon_{\mathcal{A}}.$$

The reduction loses a factor of l. Note that l is a fixed, polynomially bounded positive integer independent of the number of queries made by $\mathcal{A}$, thus the reduction is tight. □

By combining Theorems 11 and 12, we get the following reduction chain (the dashed double arrow).



This completes the proof of Theorem 10.

4.4 (Sparse) Multivariate Polynomial PRFs.

Applying our expansion technique to the $(d_1, \ldots, d_m)$-Poly-style PRFs in Definition 6 yields a general class of PRFs, which we call *Multivariate Polynomial PRFs* (MP-PRFs). When the base polynomial $f(x) = a_m x^{d_m} + \cdots + a_1 x^{d_1} + a_0$ is sparse (for example, $m \leqslant 2$), we refer to the resulting constructions as *Sparse Multivariate Polynomial PRFs* (SMP-PRFs). The form of an MP-PRF (or SMP-PRF) is uniquely determined by a tuple $(f, \mathcal{S}, l)$, where f is the base polynomial, $\mathcal{S}$ is the base input space, and $l \geqslant 1$ is the expansion factor (the case $l = 1$ corresponds to the original PRFs). For base polynomials without a constant term, it suffices to consider the case $l = 1$, since the corresponding expanded PRFs coincide with those obtained by adding a constant term (see *Remark* in Section 4.3).

All existing group-action-based PRFs form a subclass of (sparse) MP-PRFs:
NR-style PRFs [2]: $(f, \mathcal{S}, l) = (a_1 x + a_0, \{0,1\}, |x|)$, $F^{\mathsf{NR}}(\mathbf{k}, x) = (a_0 + a_1 x_1 + \cdots + a_l x_l) \diamond E_0$, where $x_i \in \mathcal{S} = \{0,1\}$, and $x = x_1 \| \cdots \| x_l$. Such PRFs are secure under the GA-DDH assumption, but are restricted to $l = |x|$, which is typically set to 128 or 256. Such a large number of polynomial terms significantly limits their practicality.

Poly-style PRFs [42]: $(f, \mathcal{S}, l) = (a_d x^d + \cdots + a_1 x + a_0, \mathbb{Z}_N, 1)$, $F^{\mathsf{Poly}}(\mathbf{k}, x) = (a_d x^d + \cdots + a_1 x + a_0) \diamond E_0$. As analyzed in Theorem 4, when N is composite, such PRFs are secure only when $\mathcal{S}$ is suitably restricted to a subset of $\mathbb{Z}_N$.
Pederson's linear Poly-style PRFs [39]: $(f, \mathcal{S}, l) = (ax, \mathcal{G}, 1)$, $F^{\mathsf{LP}}(\mathbf{k}, x) = (ax) \diamond E_0$. Such PRFs are practical, but are currently only applicable to the SCALLOP parameters (large-scale group action computations under the SCALLOP setting remain relatively slow) with $N = 2q$, where q is a large prime and $\mathcal{G}$ is the subgroup of $\mathbb{Z}_N$ of order q. In addition, for linear polynomials, one must additionally assume that the adversary cannot make quantum queries to the PRF evaluation oracle; otherwise, the secret key can be efficiently recovered.

Our Choice of the Tuple $(f, \mathcal{S}, l)$. We adopt the two families of Sparse-Poly PRFs in Table 3 as the base PRFs, and set the expansion factor to $l = \lceil |x| / \log_2 |\mathcal{S}| \rceil$ according to the target input length $|x|$. The resulting SMP-PRFs are of the following form:

$$F^{\mathsf{SMP}}(\mathbf{k}, x) = (c + a_1 x_1^2 + \cdots + a_l x_l^2) \diamond E_0, \tag{16}$$

where the input $x = x_1 \| \ldots \| x_l \in \mathcal{S}^l$ with $x_i \in \mathcal{S}$ for all $i \in [l]$, and the evaluation key $\mathbf{k} = (c, a_1, \ldots, a_l) \xleftarrow{\$} \mathbb{Z}_N^{l+1}$. The other family of SMP-PRFs is of the following form:

$$F^{\mathsf{SMP}}(\mathbf{k}, x) = (c + a_1 x_1 + \cdots + a_l x_l + b_1 x_1^d + \cdots + b_l x_l^d) \diamond E_0,$$

where the input $x = x_1 \| \ldots \| x_l \in \mathcal{S}^l$ with $x_i \in \mathcal{S}$ for all $i \in [l]$, and the evaluation key $\mathbf{k} = (c, a_1, \ldots, a_l, b_1, \ldots, b_l) \xleftarrow{\$} \mathbb{Z}_N^{2l+1}$. Here, the valid input space $\mathcal{S} \subset \mathbb{Z}_N$ of both SMP-PRFs should be carefully chosen according to the upper bound established in Theorem 5. In Section 4.2, we provide a strategy for selecting $\mathcal{S}$ for general composite N.

5 The Proposed VRFs

Compared with PRFs, Verifiable Random Functions (VRFs) [33] produce an additional proof of output correctness, which convinces any verifier that the output is correctly evaluated on a claimed input x under a committed evaluation key k. In this section, we first construct an interactive proof of correctness based on a Sigma protocol and then transform it into a non-interactive proof system using the Fiat–Shamir transform. Our Sigma protocol (our definition of Sigma protocol includes correctness) construction has *unpredictable commitments* and satisfies *special soundness* and *special honest-verifier zero knowledge (special HVZK)*. These properties ensure that the resulting non-interactive proof system satisfies *soundness* and *zero knowledge*, both of which are essential for the security proofs of our VRF scheme. We adopt the definitions of Sigma protocols, non-interactive proof systems, and the aforementioned security properties given in Sections 19.4 and 20.3 of [10] (Version 0.6, January 2023). For conciseness, we do not repeat these definitions here.

Next, we present our practical VRF scheme based on the SMP-PRFs (16) introduced in Section 4.4. VRFs based on other SMP-PRFs can be constructed following the same methodology.

Key Generation. Instead of directly using an encryption or commitment scheme [14], we use pseudorandom values of the PRF evaluated at several fixed inputs as the verification key, where these fixed inputs are made public as part of the public parameters. In exchange, the corresponding VRF input space must exclude these constant-many fixed inputs, which has a negligible impact on its size. This approach yields a clean and direct security proof while, in the group-action setting, halving the public-key size compared with using a commitment scheme from group actions [14].

Given a CEGA$^{\mathsf{T}}$ $(\mathbb{Z}_N, \mathcal{E}, \diamond, E_0)$ and the SMP-PRFs (16), we define the verification key as $\mathbf{vk} = (V_0, \ldots, V_l) = \big(F^{\mathsf{SMP}}(\mathbf{k}, u^{(0)}), \ldots, F^{\mathsf{SMP}}(\mathbf{k}, u^{(l)})\big) \in \mathcal{E}^{l+1}$. Note that we require at least $l+1$ pseudorandom values on distinct inputs to ensure that no two distinct PRF keys generate the same verification key. To ensure this binding property, we further require the following matrix

$$\mathbf{T} = \begin{pmatrix} 1 & (u_1^{(0)})^2 & \cdots & (u_l^{(0)})^2 \\ 1 & (u_1^{(1)})^2 & \cdots & (u_l^{(1)})^2 \\ \vdots & \vdots & & \vdots \\ 1 & (u_1^{(l)})^2 & \cdots & (u_l^{(l)})^2 \end{pmatrix}, \tag{17}$$

to be invertible over the ring $\mathbb{Z}_N$, where $u^{(i)} = u_1^{(i)} \| \ldots \| u_l^{(i)}$ is the i-th fixed input. We present a simple and efficient way to achieve this: set $u^{(0)} = 0\|0\| \ldots \|0$, $u^{(1)} = 1\|0\| \ldots \|0$, $u^{(2)} = 0\|1\| \ldots \|0$, $\ldots$, and $u^{(l)} = 0\|0\| \ldots \|1$. Consequently, the verification key is $(c \diamond E_0, (c + a_1) \diamond E_0, \cdots, (c + a_l) \diamond E_0)$. Indeed, these $l+1$ pseudorandom values can be viewed as additionally granting the adversary $l+1$ PRF evaluation-oracle queries. Therefore, by relying only on the pseudorandomness of the underlying PRFs and the zero-knowledge property of the proof system, the (residual) pseudorandomness property of our VRFs follows naturally.

Zero-Knowledge Proof of Correctness. As discussed above, let $(u^{(0)}, u^{(1)}, \ldots, u^{(l)})$ be $l+1$ distinct inputs, where $u^{(i)} = u_1^{(i)} \| \ldots \| u_l^{(i)}$, such that the matrix $\mathbf{T}$ in (17) is invertible. Then, for a PRF key $\mathbf{k} = (c, a_1, \ldots, a_l)$, the corresponding verification key is $\mathbf{vk} = (V_0, \ldots, V_l) = \big((c + a_1(u_1^{(0)})^2 + \cdots + a_l(u_l^{(0)})^2) \diamond E_0, \ldots, (c + a_1(u_1^{(l)})^2 + \cdots + a_l(u_l^{(l)})^2) \diamond E_0\big)$. To prove the correctness of an input-output pair (x, Y), where $x = x_1 \| \ldots \| x_l$, the prover equivalently proves the following relation $\mathcal{R}$:

$$\mathcal{R} = \left\{ (\mathsf{pp}, \mathbf{vk}, x, Y); \mathbf{k} \;\middle|\; \begin{array}{c} Y = (c + a_1 x_1^2 + \cdots + a_l x_l^2) \diamond E_0 \quad \wedge \\ V_i = (c + a_1(u_1^{(i)})^2 + \cdots + a_l(u_l^{(i)})^2) \diamond E_0 \quad \text{for all } i \in [0, l] \end{array} \right\} \tag{18}$$

where $\mathsf{pp} = ((\mathbb{Z}_N, \mathcal{E}, \diamond, E_0), l, \{u^{(i)}\}_{i=0}^{l})$.

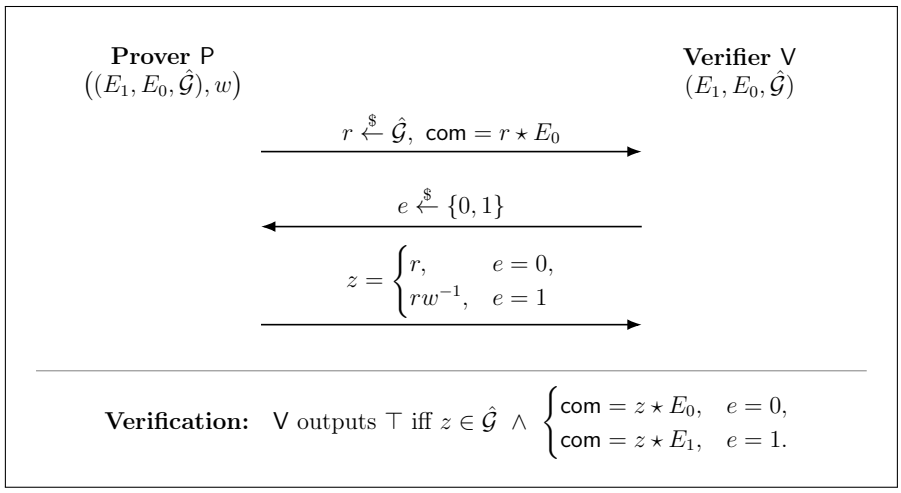
Our core idea is to transform the relation $\mathcal{R}$ into a *subgroup membership relation*.

Subgroup Membership Proof in Group Action. Formally, given an effective regular group action $(\mathcal{G}, \mathcal{E}, \star, E_0)$[17], a subgroup membership proof for group actions aims to prove that a statement $(E_1, E_0, \hat{\mathcal{G}})$ belongs to the language defined by the following relation

$$\mathcal{R}_{\hat{\mathcal{G}}} = \left\{ ((E_1, E_0, \hat{\mathcal{G}}), w) \;\middle|\; E_1 = w \star E_0 \wedge w \in \hat{\mathcal{G}} \right\},$$

where $E_1, E_0 \in \mathcal{E}$, and $\hat{\mathcal{G}}$ is a subgroup of $\mathcal{G}$. We require that there exist efficient algorithms for uniformly sampling elements from $\hat{\mathcal{G}}$ and deciding membership in $\hat{\mathcal{G}}$. The above relation $\mathcal{R}_{\hat{\mathcal{G}}}$ can be proved using a simple Sigma protocol satisfying correctness, special soundness, and honest-verifier zero-knowledge (HVZK), as shown in Fig. 3. We do not prove these properties here. Instead, we will directly prove the security properties of the Sigma protocol constructed later for our target relation $\mathcal{R}$.

Fig. 3. Sigma protocol for Subgroup Membership Relation $\mathcal{R}_{\hat{\mathcal{G}}}$.



The subgroup-membership proof in Fig. 3 is useful for proving relations over group actions. By appropriately defining a group $\mathcal{G}$, a subgroup $\hat{\mathcal{G}}$, and a group action $\star$, many seemingly complicated relations can be cast as special instances of the subgroup-membership proof. This observation is the central idea underlying our treatment of the relation $\mathcal{R}$. The subgroup-membership proof or its variants can be found in many works, such as [14,28,42].

Transform the relation $\mathcal{R}$ into a subgroup membership relation. Given a CEGA$^{\mathsf{T}}$ $(\mathbb{Z}_N, \mathcal{E}, \diamond, E_0)$, we define a new group action $\diamond' : \mathbb{Z}_N^n \times \mathcal{E}^n \to \mathcal{E}^n$ by $(a_1, \ldots, a_n) \diamond' (E_1, \ldots, E_n) := (a_1 \diamond E_1, \ldots, a_n \diamond E_n)$. Here, the group operation "+" on $\mathbb{Z}_N^n$ is

[17] Here, for the sake of generality, we consider the general EGA model. Note that the group $\mathcal{G}$ is written multiplicatively and is not necessarily abelian.

defined componentwise as $(a_1, \ldots, a_n) + (b_1, \ldots, b_n) := (a_1 + b_1, \ldots, a_n + b_n)$. It is easy to verify that $\diamond'$ indeed defines a regular group action of $(\mathbb{Z}_N^n, +)$ on $\mathcal{E}^n$. In particular, when $E_1 = \cdots = E_n = E$, we simply write $(a_1, \ldots, a_n) \diamond' E := (a_1 \diamond E, \ldots, a_n \diamond E)$.

Next, we transform the relation $\mathcal{R}$ into a subgroup membership relation. Let $v_i = c + a_1(u_1^{(i)})^2 + \cdots + a_l(u_l^{(i)})^2 (i \in [0, l])$, and $y = c + a_1 x_1^2 + \cdots + a_l x_l^2$. From (17), it follows that

$$\mathbf{T} \begin{pmatrix} c \\ a_1 \\ \vdots \\ a_l \end{pmatrix} = \begin{pmatrix} 1 & (u_1^{(0)})^2 & \cdots & (u_l^{(0)})^2 \\ 1 & (u_1^{(1)})^2 & \cdots & (u_l^{(1)})^2 \\ \vdots & \vdots & & \vdots \\ 1 & (u_1^{(l)})^2 & \cdots & (u_l^{(l)})^2 \end{pmatrix} \begin{pmatrix} c \\ a_1 \\ \vdots \\ a_l \end{pmatrix} = \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_l \end{pmatrix}. \tag{19}$$

Since $\mathbf{T}$ is invertible, we further obtain

$$y = \begin{pmatrix} 1 & x_1^2 & \cdots & x_l^2 \end{pmatrix} \begin{pmatrix} c \\ a_1 \\ \vdots \\ a_l \end{pmatrix} = \begin{pmatrix} 1 & x_1^2 & \cdots & x_l^2 \end{pmatrix} \mathbf{T}^{-1} \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_l \end{pmatrix}.$$

Let $\begin{pmatrix} t_0 & t_1 & \cdots & t_l \end{pmatrix} = \begin{pmatrix} 1 & x_1^2 & \cdots & x_l^2 \end{pmatrix} \mathbf{T}^{-1}$. We obtain the following linear relation between y and $(v_0, \ldots, v_l)$:

$$y = t_0 v_0 + t_1 v_1 + \cdots + t_l v_l. \tag{20}$$

Define $\hat{\mathcal{G}} = \{(z_0, z_1, \ldots, z_l, \tilde{z}) \in \mathbb{Z}_N^{l+2} \mid \tilde{z} = \sum_{i=0}^{l} t_i z_i\}$, which can be easily verified to be a subgroup of $(\mathbb{Z}_N^{l+2}, +)$. Then, the relation $\mathcal{R}$ can be equivalently written as the following subgroup membership relation

$$\mathcal{R}' = \left\{ (\mathsf{pp}, \mathbf{vk}, x, Y); \mathbf{w} \;\middle|\; (V_0, V_1, \ldots, V_l, Y) = \mathbf{w} \diamond' E_0 \;\wedge\; \mathbf{w} \in \hat{\mathcal{G}} \right\}. \tag{21}$$

where $\mathbf{w} = (v_0, v_1, \ldots, v_l, y)$. Moreover, in the isogeny-based group action setting, quadratic twists enable efficient verification for the negative challenge $e = -1$. When N is odd, we therefore extend the challenge space from $\{0, 1\}$ to $\{-1, 0, 1\}$. Since 2 is invertible in $\mathbb{Z}_N$, any two accepting conversations with the same commitment and distinct challenges allow extraction of the witness. When N is even, we retain the binary challenge space $\{0, 1\}$, since the challenges 1 and -1 would otherwise yield only $2\mathbf{w} \in \hat{\mathcal{G}}$, which does not necessarily imply $\mathbf{w} \in \hat{\mathcal{G}}$. For odd N, we present our ternary-challenge Sigma protocol for the relation $\mathcal{R}$ in Fig. 4. The standard $\{0, 1\}$-challenge version is obtained simply by removing the case $e = -1$, and hence we do not describe it separately. Note that, unlike the framework in Fig. 3, the prover's response $\mathbf{z}$ in Fig. 4 contains only the randomness that determines the corresponding group element, rather than the group element itself. Accordingly, the verifier can reconstruct the group element from this randomness, thereby eliminating the need for a subgroup membership check. This further reduces the response size by $\log_2 N$.

Fig. 4. Sigma Protocol for Relation $\mathcal{R}$

Round 1: $\mathsf{P}_1((\mathsf{pp}, \mathbf{vk}, x, Y), \mathbf{k})$
Compute $\mathbf{t} = \left(1\ x_1^2 \cdots x_l^2\right) \mathbf{T}^{-1}$
For $i = 1$ to $l+1$ do
  $t_{i-1} \leftarrow \mathbf{t}[i]$
Sample $\mathbf{r} = (r_0, \ldots, r_l) \xleftarrow{\$} \mathbb{Z}_N^{l+1}$
Compute $\mathbf{r}' \leftarrow (r_0, \ldots, r_l, t_0 r_0 + \cdots + t_l r_l)$
$\mathsf{com} \leftarrow \mathbf{r}' \diamond' E_0$
Prover sends com to verifier

Round 2: $\mathsf{V}_1((\mathsf{pp}, \mathbf{vk}, x, Y), \mathsf{com})$
$e \xleftarrow{\$} \{-1, 0, 1\}$
Verifier sends e to prover

Round 3: $\mathsf{P}_2((\mathsf{pp}, \mathbf{vk}, x, Y), \mathbf{k}, \mathsf{com}, e)$
Compute $\mathbf{v} \leftarrow \mathbf{T}\mathbf{k}$, $\mathbf{z} \leftarrow \mathbf{r} - e\mathbf{v}$
Prover sends $\mathbf{z}$ to verifier

Verify: $\mathsf{V}_2((\mathsf{pp}, \mathbf{vk}, x, Y), \mathsf{com}, e, \mathbf{z})$
For $i = 1$ to $l+1$ do
  $z_{i-1} \leftarrow \mathbf{z}[i]$
Compute $\mathbf{t} = \left(1\ x_1^2 \cdots x_l^2\right) \mathbf{T}^{-1}$
For $i = 1$ to $l+1$
  do $t_{i-1} \leftarrow \mathbf{t}[i]$
Compute $\mathbf{z}' \leftarrow (z_0, \ldots, z_l, t_0 z_0 + \cdots + t_l z_l)$
If $e = 0$ then compute
  $\mathsf{com}' \leftarrow \mathbf{z}' \diamond' E_0$
Else if $e = 1$ then compute
  $\mathsf{com}' \leftarrow \mathbf{z}' \diamond' (V_0, \ldots, V_l, Y)$
Else compute
  $\mathsf{com}' \leftarrow \mathbf{z}' \diamond' (V_0, \ldots, V_l, Y)^\top$
Output $\mathsf{com} == \mathsf{com}'$

Remark: In *Round 3*, $\mathbf{v} := (v_0, \ldots, v_l)$. The operator "$==$" denotes an equality test. It outputs 1 if the two operands are equal, and outputs 0 otherwise. We write $(V_0, \ldots, V_l, Y)^\top := (V_0^\top, \ldots, V_l^\top, Y^\top)$ for brevity.

In what follows, we prove the correctness and security properties of the ternary-challenge Sigma protocol under the assumption that N is odd. The corresponding proof for the binary-challenge version with challenge space $\{0, 1\}$ is analogous, so we do not repeat it here. Note that the binary-challenge version applies to arbitrary N and does not require the underlying group action to support a twist operation.

Theorem 13 *For a* CEGA$^\mathsf{T}$ $(\mathbb{Z}_N, \mathcal{E}, \diamond, E_0)$ *with odd* N*, the protocol in Fig. 4 is a correct Sigma protocol, has unpredictable commitments, and satisfies special soundness and special honest-verifier zero knowledge (special HVZK).*

Proof. *Correctness for Sigma protocol.* To prove correctness, we show that, for any $(\mathsf{st} = (\mathsf{pp}, \mathbf{vk}, x, Y), \mathsf{wt} = \mathbf{k}) \in \mathcal{R}$, the conversation $(\mathsf{com}, e, \mathbf{z})$ generated by an honest execution of P_1, V_1, and P_2 in Fig. (4) is always accepted by V_2, i.e., V_2 outputs 1.

– If $e = 0$, then $\mathbf{z} = \mathbf{r} = (r_0, \ldots, r_l)$. Hence,

$$\mathbf{z}' = (r_0, \ldots, r_l, t_0 r_0 + \cdots + t_l r_l) = \mathbf{r}'.$$

Consequently, $\mathbf{z}' \diamond' E_0 = \mathbf{r}' \diamond' E_0 = \mathsf{com}$, which implies that V_2 outputs 1 (accept).

– If $e = 1$, then $\mathbf{z} = (r_0 - v_0, \ldots, r_l - v_l)$. Hence, we have

$$\mathbf{z}' = (r_0 - v_0, \ldots, r_l - v_l, t_0(r_0 - v_0) + \cdots + t_l(r_l - v_l)).$$

Since $y = t_0 v_0 + t_1 v_1 + \cdots + t_l v_l$ by Equation (20), we have

$$\begin{aligned}
&\mathbf{z}' \diamond' (V_0, \ldots, V_l, Y) \\
=&(r_0 - v_0 + v_0, \ldots, r_l - v_l + v_l, (t_0 r_0 + \cdots + t_l r_l) - (t_0 v_0 + \cdots + t_l v_l) + y) \diamond' E_0 \\
=&(r_0, \ldots, r_l, t_0 r_0 + \cdots + t_l r_l) \diamond' E_0 = \mathbf{r}' \diamond' E_0.
\end{aligned}$$

Thus, we have $\mathbf{z}' \diamond' (V_0, \ldots, V_l, Y) = \mathbf{r}' \diamond' E_0 = \mathsf{com}$, which implies that V_2 outputs 1 (accept).

If $e = -1$, then $\mathbf{z} = (r_0 + v_0, \ldots, r_l + v_l)$ and thus

$$\mathbf{z}' = (r_0 + v_0, \ldots, r_l + v_l, t_0(r_0 + v_0) + \cdots + t_l(r_l + v_l))$$

Similarly, by Equation (20), we have

$$\begin{aligned}
&\mathbf{z}' \diamond' (V_0, \ldots, V_l, Y)^\top \\
=&(r_0 + v_0 - v_0, \ldots, r_l + v_l - v_l, (t_0 r_0 + \cdots + t_l r_l) + (t_0 v_0 + \cdots + t_l v_l) - y) \diamond' E_0 \\
=&(r_0, \ldots, r_l, t_0 r_0 + \cdots + t_l r_l) \diamond' E_0 = \mathbf{r}' \diamond' E_0.
\end{aligned}$$

Thus, we have $\mathbf{z}' \diamond' (V_0, \ldots, V_l, Y)^\top = \mathbf{r}' \diamond' E_0 = \mathsf{com}$, which implies that V_2 outputs 1 (accept).

Special soundness. Given a statement $\mathsf{st} = (\mathsf{pp}, \mathbf{vk}, x, Y)$ and two valid conversations $(\mathsf{com}, e_1, \mathbf{z}_1)$ and $(\mathsf{com}, e_2, \mathbf{z}_2)$ with $e_1 \neq e_2$, we construct the following extraction algorithm Ext to extract the witness wt for the statement st. Since $e_1, e_2 \in \{-1, 0, 1\}$ and $e_1 \neq e_2$, at least one of the following equations must hold for $(i, j) = (1, 2)$ or $(2, 1)$.

$$\begin{aligned}
&\mathbf{z}'_i \diamond' E_0 = \mathbf{z}'_j \diamond' (V_0, \ldots, V_l, Y) \\
&\mathbf{z}'_i \diamond' E_0 = \mathbf{z}'_j \diamond' (V_0^\top, \ldots, V_l^\top, Y^\top) \\
&\mathbf{z}'_i \diamond' (V_0, \ldots, V_l, Y) = \mathbf{z}'_j \diamond' (V_0^\top, \ldots, V_l^\top, Y^\top)
\end{aligned}$$

For the first two of these three cases, the algorithm Ext extracts $(v_0, \ldots, v_l)$ directly by computing $\pm(\mathbf{z}_i - \mathbf{z}_j)$. For the last case, Ext first extracts $(\gamma_0, \ldots, \gamma_l) = (2v_0, \ldots, 2v_l)$ by computing $\mathbf{z}_j - \mathbf{z}_i$, then it can compute $(v_0, \ldots, v_l)$ directly, since 2 is invertible in $\mathbb{Z}_N$. After extracting $(v_0, \ldots, v_l)$, Ext solves the following system of linear equations over $\mathbb{Z}_N$,

$$\begin{cases}
v_0 = c + a_1 (u_1^{(0)})^2 + \cdots + a_l (u_l^{(0)})^2 \\
v_1 = c + a_1 (u_1^{(1)})^2 + \cdots + a_l (u_l^{(1)})^2 \\
\vdots \\
v_l = c + a_1 (u_1^{(l)})^2 + \cdots + a_l (u_l^{(l)})^2
\end{cases}$$

noticing that the coefficient matrix is $\mathbf{T}$ (17) is known in pp and invertible over $\mathbb{Z}_N$, we can gets the unique witness

$$\mathsf{wt} = \begin{pmatrix} c \\ a_1 \\ \vdots \\ a_l \end{pmatrix} = \mathbf{T}^{-1} \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_l \end{pmatrix}.$$

Using the vectors $\mathbf{z}'_i, \mathbf{z}'_j \in \hat{\mathcal{G}}$ reconstructed by the verifier, the extractor computes

$$\mathbf{w} = (e_i - e_j)^{-1}(\mathbf{z}'_j - \mathbf{z}'_i) = (v_0, \ldots, v_l, y) \in \hat{\mathcal{G}},$$

where the inverse exists because N is odd. Hence, $y = \sum_{i=0}^{l} t_i v_i$ and $(V_0, \ldots, V_l, Y) = \mathbf{w} \diamond' E_0$. Since $\mathbf{k} = \mathbf{T}^{-1}(v_0, \ldots, v_l)^{\mathsf{T}}$, these equalities imply that $\mathbf{k}$ satisfies all the equations in (18). Therefore, $\mathsf{wt} = \mathbf{k}$ is a valid witness for st.

Special HVZK. We construct a simulator Sim that, on input $(\mathsf{st} = (\mathsf{pp}, \mathbf{vk}, x, Y), e)$, where $e \in \{-1, 0, 1\}$, outputs a commitment-response pair $(\mathsf{com}, \mathbf{z})$. Specifically, Sim first samples $\mathbf{z} = (z_0, \ldots, z_l) \xleftarrow{\$} \mathbb{Z}_N^{l+1}$ uniformly at random, and then computes com' following the same procedure as V_2 for the corresponding challenge e. Finally, it sets $\mathsf{com} = \mathsf{com}'$ and outputs $(\mathsf{com}, \mathbf{z})$. For every fixed $e \in \{-1, 0, 1\}$, the simulated transcript has exactly the same distribution as an honest transcript conditioned on the verifier's challenge being e. Consequently, when $e \xleftarrow{\$} \{-1, 0, 1\}$, the joint distribution of $(\mathsf{com}, e, \mathbf{z})$ is identical to that of an honest transcript.

Unpredictable commitments. For any fixed $(\mathsf{st} = (\mathsf{pp}, \mathbf{vk}, x, Y), \mathsf{wt} = \mathbf{k}) \in \mathcal{R}$ and any fixed commitment $\widehat{\mathsf{com}} = (s_0, \ldots, s_l, \hat{s}) \diamond' E_0$ in the commitment space, where $\hat{s} = t_0 s_0 + \cdots + t_l s_l$, the algorithm P_1 in Fig. 4 generates the commitment $\mathsf{com} = (r_0, \ldots, r_l, \hat{r}) \diamond' E_0$, where $\hat{r} = t_0 r_0 + \cdots + t_l r_l$ and $(r_0, \ldots, r_l)$ is sampled uniformly at random from $\mathbb{Z}_N^{l+1}$. Since the group action $\diamond'$ is regular, we have

$$\Pr[\mathsf{com} = \widehat{\mathsf{com}}] = \Pr[(r_0, \ldots, r_l) = (s_0, \ldots, s_l)] = \frac{1}{N^{l+1}},$$

which is negligible in λ. □

To make the soundness error negligible, we repeat the Sigma protocol in Fig. 4 in parallel κ times, where $3^\kappa \approx 2^\lambda$. The resulting parallel-repeated Sigma protocol has unpredictable commitments, is sound,[18] and satisfies special HVZK. Furthermore, our parallel-repeated protocol has *backward computable commitments* [10], that is, given a statement st, a challenge vector $e = (e_1, \ldots, e_\kappa)$, and a response vector $\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_\kappa)$, there exists a unique commitment com such that $(\mathsf{com}, e, \mathbf{z})$ is an accepting conversation, and the verifier can efficiently reconstruct this unique commitment vector $\mathsf{com} = (\mathsf{com}_1, \ldots, \mathsf{com}_\kappa)$ by applying

[18] By special soundness and Theorem 20.1 of [10], soundness follows when the challenge space is exponentially large.

the reconstruction procedure in V_2 componentwise. This property allows us to apply the optimized Fiat–Shamir transform [10] to the parallel-repeated Sigma protocol. In this optimized transform, a proof contains only the challenge and the response, while the commitment is reconstructed during verification. The resulting non-interactive proof system is presented in Fig. 5. By Lemma 14, derived from Theorems 20.2 and 20.3 and Exercise 20.14 in Version 0.6 (January 2023) of [10], together with Theorem 13, the non-interactive proof system in Fig. 5 is both sound and zero knowledge in the random-oracle model (Theorem 15).

Fig. 5. Non-interactive proof system for relation $\mathcal{R}$

Prover $\mathsf{P}(\mathsf{st} = (\mathsf{pp}, \mathbf{vk}, x, Y), \mathsf{wt} = \mathbf{k})$
For $i = 1$ to κ do
  $\mathsf{com}_i \leftarrow \mathsf{P}_1(\mathsf{st}, \mathsf{wt})$
$\mathsf{com} \leftarrow (\mathsf{com}_1, \ldots, \mathsf{com}_\kappa)$
$e = e_1 \| \ldots \| e_\kappa \leftarrow \mathcal{H}(\mathsf{FS} \parallel \mathsf{st} \parallel \mathsf{com})$
For $i = 1$ to κ do
  $\mathbf{z}_i \leftarrow \mathsf{P}_2(\mathsf{st}, \mathsf{wt}, \mathsf{com}_i, e_i)$
$\mathbf{z} \leftarrow (\mathbf{z}_1, \ldots, \mathbf{z}_\kappa)$
Output $\pi = (e, \mathbf{z})$

Verifier $\mathsf{V}(\mathsf{st}, \pi)$
Expand $\mathbf{z}$ to $(\mathbf{z}_1, \ldots, \mathbf{z}_\kappa)$
Compute $\mathbf{t} = \begin{pmatrix} 1 & x_1^2 & \cdots & x_l^2 \end{pmatrix} \mathbf{T}^{-1}$
For $i = 1$ to $l + 1$ do
  $t_{i-1} \leftarrow \mathbf{t}[i]$
For $i = 1$ to κ do
  For $j = 1$ to $l + 1$ do
    $z_{i,j-1} \leftarrow \mathbf{z}_i[j]$
  compute $\mathbf{z}'_i \leftarrow (z_{i,0}, \ldots, z_{i,l}, t_0 z_{i,0} + \cdots + t_l z_{i,l})$
  If $e_i = 0$ then
    compute $\mathsf{com}'[i] \leftarrow \mathbf{z}'_i \diamond' E_0$
  Else if $e_i = 1$ then
    compute $\mathsf{com}'[i] \leftarrow \mathbf{z}'_i \diamond' (V_0, \ldots, V_l, Y)$
  Else
    compute $\mathsf{com}'[i] \leftarrow \mathbf{z}'_i \diamond' (V_0, \ldots, V_l, Y)^\top$
Output $e == \mathcal{H}(\mathsf{FS} \parallel \mathsf{st} \parallel \mathsf{com}')$

Remark: Here, $3^\kappa \approx 2^\lambda$. $\mathcal{H}(\mathsf{FS}\| \cdot)$ is a hash function whose output space is $\{-1, 0, 1\}^\kappa$. $\mathsf{P}_1, \mathsf{P}_2$ are the prover algorithm in Fig 4.

Lemma 14 *Let $\Pi_\Sigma = (\mathcal{P}, \mathcal{V})$ be a correct Sigma protocol for a relation $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{Y}$ that has unpredictable and backward-computable commitments, and satisfies soundness and special honest-verifier zero knowledge. Let $\mathsf{FS}\text{-}\Pi_\Sigma$ be the non-interactive proof system obtained by applying the (optimized) Fiat–Shamir transform to Π_Σ using a hash function $\mathcal{H}$. If $\mathcal{H}$ is modeled as a random oracle, then $\mathsf{FS}\text{-}\Pi_\Sigma$ is a non-interactive proof system for $\mathcal{R}$ that satisfies soundness and zero knowledge.*

Theorem 15 *For a* CEGA$^\mathsf{T}$ $(\mathbb{Z}_N, \mathcal{E}, \diamond, E_0)$ *with odd N, the non-interactive proof system (Fig. 5) for relation $\mathcal{R}$ satisfies correctness, non-interactive soundness, and zero knowledge under the random oracle model.*

Description of VRFs. The above ingredients suffice to present our VRF construction.

$\mathsf{VRF.Setup}(1^\lambda) \to \mathsf{pp}$. Given the security parameter λ, this algorithm outputs the public parameter $\mathsf{pp} = ((\mathbb{Z}_N, \mathcal{E}, \diamond, E_0), (f, \mathcal{S}, l), \{u^{(i)}\}_{i=0}^l)$, where $(\mathbb{Z}_N, \mathcal{E}, \diamond, E_0)$ is a CEGA$^\mathsf{T}$, $(f, \mathcal{S}, l)$ is the parameter determining the structure of the SMP-PRFs, and $u^{(0)}, \ldots, u^{(l)}$ are $l+1$ fixed input values in $\mathcal{S}$ such that the matrix $\mathbf{T}$ in (17) is invertible.

$\underline{\mathsf{VRF.KeyGen}(\mathsf{pp}) \to (\mathbf{k}, \mathbf{vk})}$. Given the public parameter pp, this algorithm randomly chooses a PRF key $\mathbf{k} = (c, a_1, \ldots, a_l) \xleftarrow{\$} \mathbb{Z}_N^{l+1}$, computes $V_i \leftarrow (c + a_1(u_1^{(i)})^2 + \cdots + a_l(u_l^{(i)})^2) \diamond E_0$ for every $i \in [0, l]$, sets $\mathbf{vk} \leftarrow (V_0, V_1, \ldots, V_l)$, and outputs $(\mathbf{k}, \mathbf{vk})$.

$\underline{\mathsf{VRF.Eval}(\mathsf{pp}, \mathbf{k}, \mathbf{vk}, x) \to (Y, \pi)}$. Given the public parameter pp, a key pair $(\mathbf{k}, \mathbf{vk}) \in \mathbb{Z}_N^{l+1} \times \mathcal{E}^{l+1}$, and an input value $x = x_1 \| \ldots \| x_l \in \mathcal{S}^l \setminus \{u^{(i)}\}_{i=0}^l$ $(x_i \in \mathcal{S})$, this algorithm computes the evaluation $Y \leftarrow (c + a_1 x_1^2 + \cdots + a_l x_l^2) \diamond E_0$ and the proof $\pi \leftarrow \mathsf{P}(\mathsf{st} = (\mathsf{pp}, \mathbf{vk}, x, Y), \mathsf{wt} = \mathbf{k})$, where P is the prover algorithm in Fig. 5, and outputs (Y, π). Note that if $x \notin \mathcal{S}^l \setminus \{u^{(i)}\}_{i=0}^l$, the algorithm outputs $\perp$.

$\underline{\mathsf{VRF.Verify}(\mathsf{pp}, \mathbf{vk}, x, Y, \pi) \to 0 \text{ or } 1}$. Given the public parameter pp, a verification key $\mathbf{vk}$, an input x, an output Y, and a proof π, this algorithm returns 0 if $x \notin S^l \setminus \{\mathbf{u}^{(i)}\}_{i=0}^l$. Otherwise, it computes $b \leftarrow \mathsf{V}(\mathsf{st} = (\mathsf{pp}, \mathbf{vk}, x, Y), \pi)$ and returns b, where V is the verifier algorithm in Fig. 5.

Theorem 16 *For a* CEGA$^{\mathsf{T}}$ $(\mathbb{Z}_N, \mathcal{E}, \diamond, E_0)$ *with odd* N*, our VRF construction satisfies complete provability, unique provability, and (residual) pseudorandomness in the* GGAM$^{\mathsf{T}}$ *and random oracle models under the* GA-DDH *assumption.*

Proof. *Complete provability:* Note that, for any tuple $(\mathsf{pp}, \mathbf{sk}, \mathbf{vk}, x, Y, \pi)$ generated according to $\mathsf{pp} \leftarrow \mathsf{VRF.Setup}(1^\lambda)$, $(\mathbf{sk}, \mathbf{vk}) \leftarrow \mathsf{VRF.KeyGen}(\mathsf{pp})$, $(Y, \pi) \leftarrow \mathsf{VRF.Eval}(\mathsf{pp}, \mathbf{sk}, \mathbf{vk}, x)$, $(\mathsf{st} = (\mathsf{pp}, \mathbf{vk}, x, Y), \mathsf{wt} = \mathbf{sk}) \in \mathcal{R}$ (18), namely, $(\mathsf{pp}, \mathbf{vk}, x, Y) \in \mathcal{L}_\mathcal{R}$. Since the algorithm $\mathsf{VRF.Verify}$ follows the verifier algorithm in our non-interactive proof system of relation $\mathcal{R}$, the complete provability is held for the correctness (by Theorem 15) of our non-interactive proof system.

Unique provability. In unique provability game, a PPT adversary $\mathcal{A}$ wins the game if it find out two pairs $(x, Y, \pi), (x, Y', \pi')$ such that

$$\mathsf{VRF.Verify}(\mathsf{pp}, \mathbf{vk}, x, Y, \pi) = \mathsf{VRF.Verify}(\mathsf{pp}, \mathbf{vk}, x, Y', \pi') = 1 \wedge Y \neq Y'.$$

However, for any key pair $(\mathbf{k}, \mathbf{vk})$ that adversary chooses, one fact remains immutable, namely, at least one of this two statements $\mathsf{st} = (\mathsf{pp}, \mathbf{vk}, x, Y)$, $\mathsf{st}' = (\mathsf{pp}, \mathbf{vk}, x, Y')$ is not in $\mathcal{L}_\mathcal{R}$, i.e., at least one of this two statements is false statement. So we have

$$\mathbf{AdvVRF}^{\mathsf{UP}}_{\mathcal{A}, \Pi_{\mathsf{VRF}}}(\lambda) \leqslant \Pr[\mathcal{A} \text{ generates a valid proof for } \mathsf{st} \notin \mathcal{L}_\mathcal{R}]$$

Following Theorem 15, our non-interactive proof system is sound for PPT adversary when $\mathcal{H}$ is modeled as a random oracle, which means $\mathbf{AdvVRF}^{\mathsf{UP}}_{\mathcal{A}, \Pi_{\mathsf{VRF}}}(\lambda)$ is also negligible.

(Residual) pseudorandomness. We show that the (Residual) pseudorandomness of our VRFs reduces to the pseudorandomness of the underlying SMP-PRFs (16)

$$F^{\mathsf{SMP}}(\mathbf{k}, x = x_1 \| \ldots \| x_l) = (c + a_1 x_1^2 + \cdots + a_l x_l^2) \diamond E_0.$$

Here, we use the equivalent definition 2 of pseudorandomness introduced in Section 2.2. Assuming that there exists a PPT algorithm $\mathcal{A}$ that can break the (residual) pseudorandomness of our VRFs with non-negligible advantage $\varepsilon_{\mathcal{A}}$, we construct a PPT algorithm $\mathcal{B}$ such that $\mathcal{B}^{\mathcal{A}}$ can break the pseudorandomness of the SMP-PRFs (16) with non-negligible advantage $\varepsilon_{\mathcal{B}} = \varepsilon_{\mathcal{A}}(1 - \mathrm{negl}(\lambda))$, and thus $\varepsilon_{\mathcal{B}}$ is also non-negligible.

$\mathcal{B}$ is given oracle access to $\mathcal{O}_{\mathbf{k}}^{\mathsf{PR}''}$, which evaluates the PRF (16) under the key $\mathbf{k} \xleftarrow{\$} \mathbb{Z}_N^{l+1}$. Then, $\mathcal{B}$ starts to simulate the (residual) pseudorandomness experiment for $\mathcal{A}$. First, $\mathcal{B}$ chooses $l+1$ inputs $(u^{(0)}, \ldots, u^{(l)})$ such that the matrix $\mathbf{T}$ (17) is invertible. Then, it constructs the simulated verification key $\mathbf{vk}$ by querying the oracle $\mathcal{O}_{\mathbf{k}}^{\mathsf{PR}''}$ at $u^{(i)}$ for each $i \in [0, l]$, and sets

$$\mathbf{vk} \leftarrow \left(\mathcal{O}_{\mathbf{k}}^{\mathsf{PR}''}(u^{(0)}), \ldots, \mathcal{O}_{\mathbf{k}}^{\mathsf{PR}''}(u^{(l)})\right).$$

It then sends $\{u^{(i)}\}_{i=0}^{l}$, together with the remaining public parameters and the simulated verification key $\mathbf{vk}$, to $\mathcal{A}$. When $\mathcal{A}$ submits its i-th evaluation query x_i to $\mathcal{B}$, algorithm $\mathcal{B}$ queries $\mathcal{O}_{\mathbf{k}}^{\mathsf{PR}''}$ on x_i and obtains $Y_i \leftarrow \mathcal{O}_{\mathbf{k}}^{\mathsf{PR}''}(x_i)$. By Theorem 15, the NIZK proof system (P, V) described in Fig. 5 is zero knowledge. Therefore, for the statement $\mathsf{st}_i = (\mathsf{pp}, \mathbf{vk}, x_i, Y_i)$, $\mathcal{B}$ just follows the simulation procedure of the simulator Sim: it first samples $e \xleftarrow{\$} \{-1, 0, 1\}^{\kappa}$ and $\mathbf{z} \xleftarrow{\$} \left(\mathbb{Z}_N^{l+1}\right)^{\kappa}$, and then reconstructs the corresponding commitment com. Finally, it programs the random oracle as $\mathcal{H}(\mathsf{FS} \| \mathsf{st} \| \mathsf{com}) := e$, records this input-output pair in the hash table, and returns $(Y_i, \pi_i = (e, \mathbf{z}))$ to $\mathcal{A}$. If this input has already been assigned a different value, then $\mathcal{B}$ aborts. Since the newly generated commitment is unpredictable (Theorem 13), the probability that $\mathcal{A}$ has previously queried this exact hash input is negligible. By a union bound over polynomially many proof queries, $\mathcal{B}$ successfully simulates all the requested proofs with probability $1 - \mathrm{negl}(\lambda)$. Conditioned on $\mathcal{B}$ not aborting, the simulated proofs are identically distributed to honestly generated proofs. Consequently, the simulated and real views of $\mathcal{A}$ are indistinguishable.

Next, $\mathcal{A}$ submits $\hat{x}$ to $\mathcal{B}$ as its challenge input[19]. Algorithm $\mathcal{B}$ forwards $\hat{x}$ to its own challenger and receives the challenge value Z, which it then forwards to $\mathcal{A}$. After the challenge phase, $\mathcal{A}$ may issue a second round of evaluation queries. Algorithm $\mathcal{B}$ answers these queries in exactly the same way as in the first query phase. Finally, $\mathcal{A}$ outputs a guess b'. Algorithm $\mathcal{B}$ sets $b^* \leftarrow b'$ and outputs b^* as its guess in the underlying PRF pseudorandomness experiment.

Unless $\mathcal{B}$ aborts (in which case it outputs $b^* \xleftarrow{\$} \{0, 1\}$), $\mathcal{A}$'s view in the simulated execution has exactly the same distribution as its view in a real execution. Therefore, it follows that, $\varepsilon_{\mathcal{B}} = \varepsilon_{\mathcal{A}}(1 - \mathrm{negl}(\lambda))$. Consequently, if $\varepsilon_{\mathcal{A}}$ is non-negligible, then so is $\varepsilon_{\mathcal{B}}$. Finally, by Theorem 10, the underlying SMP-PRFs are pseudorandom in the $\mathsf{GGAM}^{\mathsf{T}}$ model under the GA-DDH assumption. This establishes the claimed result. □

[19] Recall that the input domain of our VRF excludes $\{u^{(i)}\}_{i=0}^{l}$; hence, $\mathcal{A}$ cannot select any $u^{(i)}$ as its challenge input.

Efficiency Costs. Table 1 provides a comparison between existing group-action-based VRFs and our VRFs, all instantiated with CSIDH-512 (where an element of $\mathbb{Z}_N$ is represented using approximately 32 bytes and a curve using 64 bytes) and approximately 128-bit soundness. Thus, the secret-key size is $32(l+1)$ bytes and verification-key size is $64(l+1)$ bytes. In each Sigma protocol repetition, the prover response contains $l+1$ elements of $\mathbb{Z}_N$, while both the prover and verifier perform $l+2$ group-action computations. Moreover, the use of quadratic twists yields the number of repetitions κ of the protocol is reduced from 128 to $128\log_3 2 \approx 80$. Hence, the proof size is $(128 \text{ bits} + 32(l+1)\cdot 80 \text{ bytes})/1024 \approx 2.5(l+1)\mathsf{KB}$, and the number of group-action computations, which is the dominant measure of computational efficiency for our VRFs, is $80(l+2)$. We use the parameter setting in Table 3 with a 94-bit input space $\mathcal{S}$. Accordingly, the expansion factor l is set to 2 and 3 for VRFs with 128-bit and 256-bit input spaces, respectively.

Optional Optimization Techniques. The following existing optional optimization techniques are applicable to further reducing the proof lengths of both the ternary-challenge and binary-challenge versions (without twists) of our VRF.

–*PRG-Based Randomness Generation.* For each parallel repetition i, instead of independently sampling the $l+1$ entries of $\mathbf{r}_i \in \mathbb{Z}_N^{l+1}$, the prover samples a short seed s_i and expands it using a secure PRG to obtain $\mathbf{r}_i$. When $e_i = 0$, the prover just send s_i instead of the full response vector; when $e_i \in \{-1, 1\}$, it sends $\mathbf{z}_i$ as usual and keeps s_i secret.

–*Fixed-Weight Challenges* [12]. Building on the above PRG-based optimization, in which a short seed s_i is used to generate $\mathbf{r}_i \in \mathbb{Z}_N^{l+1}$, one can observe that the response for $e_i = 0$ is shorter than those for $e_i \in \{-1, 1\}$. We can therefore restrict the challenge to the fixed-weight set $\mathcal{C}^{\mathsf{ter}}_{\kappa,w} := \{e \in \{-1,0,1\}^\kappa | \, \mathrm{weight}(e) = w\}$, where $\mathrm{weight}(e)$ denotes the number of nonzero entries of e. Consequently, exactly $\kappa - w$ responses can be represented by short seeds, while only w full response vectors need to be transmitted, thereby further reducing the proof size. In this case, the output space of $\mathcal{H}$ is $\mathcal{C}^{\mathsf{ter}}_{\kappa,w}$, whose cardinality is

$$|\mathcal{C}^{\mathsf{ter}}_{\kappa,w}| = 2^w \binom{\kappa}{w},$$

where κ and w should be chosen such that $2^w\binom{\kappa}{w} \approx 2^\lambda$.

Similarly, Similarly, without applying the *twists* technique, we consider the binary challenge space where $e_i \in \{0,1\}$ and define the fixed-weight set $\mathcal{C}^{\mathsf{bin}}_{\kappa,w} := \{e \in \{0,1\}^\kappa \mid \mathrm{weight}(e) = w\}$. In this case,

$$|\mathcal{C}^{\mathsf{bin}}_{\kappa,w}| = \binom{\kappa}{w},$$

where κ and w should be chosen such that $\binom{\kappa}{w} \approx 2^\lambda$.

–*Seed Trees* [12]. The κ seeds used in the above PRG-based optimization can be generated as the leaves of a binary seed tree expanded from a single root

Table 4. Expected proof sizes and group-action costs for the binary- and ternary-challenge versions under different optimization techniques.

Binary	(κ, w) **for** $l = 2, 3, 13$	$l = 2$	$l = 3$	$l = 13$	$T_{\mathsf{PG}}/T_{\mathsf{V}}$ **for** $l = 2, 3, 13$
Original	$(128, -)$	12.02 KB	16.02 KB	56.02 KB	$(512, 640, 1920)$
PRG	$(128, -)$	7.02 KB	9.02 KB	29.02 KB	$(512, 640, 1920)$
PRG + Seed Tree	$(128, -)$	6.73 KB	8.73 KB	28.73 KB	$(512, 640, 1920)$
PRG + Fixed Weight	$(165, 40); (186, 36); (306, 27)$	5.72 KB	6.86 KB	16.19 KB	$(660, 930, 4590)$
PRG + Fixed Weight + Seed Tree	$(165, 40); (186, 36); (306, 27)$	4.74 KB	5.55 KB	13.03 KB	$(660, 930, 4590)$
Ternary	(κ, w) **for** $l = 2, 3, 13$	$l = 2$	$l = 3$	$l = 13$	$T_{\mathsf{PG}}/T_{\mathsf{V}}$ **for** $l = 2, 3, 13$
Original	$(80, -)$	7.52 KB	10.02 KB	35.02 KB	$(320, 400, 1200)$
PRG	$(80, -)$	5.43 KB	7.10 KB	23.77 KB	$(320, 400, 1200)$
PRG + Seed Tree	$(80, -)$	5.36 KB	7.03 KB	23.69 KB	$(320, 400, 1200)$
PRG + Fixed Weight	$(114, 33); (131, 30); (235, 23)$	4.38 KB	5.34 KB	13.39 KB	$(456, 655, 3525)$
PRG + Fixed Weight + Seed Tree	$(114, 33); (131, 30); (235, 23)$	3.81 KB	4.53 KB	11.05 KB	$(456, 655, 3525)$

Remark: We use $\log_2 N \approx 256$*, 128-bit seeds, and* $1\,\mathrm{KB} = 1024$ *bytes.* T_{PG} *and* T_{V} *denote the number of group actions for a single proof generation or verification. The three parameter pairs* (κ, w) *in each "fixed-weight" row correspond, respectively, to* $l = 2, 3, 13$ *and are chosen to provide approximately 128 bits of challenge entropy. The selection strategy is to first minimize the proof length. If several parameter pairs achieve the same minimum length, we choose the one with the smallest* κ *to reduce the number of group-action evaluations. The reported seed-tree sizes are averaged over uniformly sampled challenges.*

seed. Once the challenge e is determined, the prover reveals only a small set of internal-node seeds that allows the verifier to recover precisely the leaf seeds corresponding to $e_i = 0$, while the leaf seeds corresponding to $e_i \in \{-1, 1\}$ in the ternary version (or $e_i = 1$ in the binary version) remain hidden. Since $\mathrm{weight}(e) = w$, this technique compresses the $\kappa - w$ individually transmitted seeds into a much smaller seed-tree opening, thereby further reducing the proof size.

The above techniques have been employed in numerous works (e.g., [6,28,12]) to reduce the lengths of proofs for group actions. In our construction, the benefits of these techniques become more pronounced as the expansion factor l increases. It should be noted, however, that while the fixed-weight-challenge technique reduces the proof size, it often increases the computational overhead, since achieving the same soundness level generally requires a larger total number of parallel repetitions. One may therefore combine these techniques flexibly according to practical requirements. In Table 4, for $l = 2, 3, 13$, we select specific parameter pairs (κ, w) and compare the resulting proof sizes under different optimization techniques. We make the following two observations. First, seed trees provide only limited improvement on average. Second, although these optimization techniques yield larger relative improvements for the binary variant, it still performs worse overall than the ternary variant. We emphasize, however, that our parameter choices are intended only as reference points. A comprehensive evaluation of practical performance requires a more detailed analysis, which is beyond the scope of this work.

6 Conclusion and Future Work

In this work, we present attacks on poly-style PRFs instantiated over composite-order class groups. To eliminate the impact of these attacks, we introduce the restrict-then-expand technique and propose a new construction of sparse MP-PRFs (SMP-PRFs). Our construction unifies existing designs while retaining efficient verifiability. We first prove the security of the base PRFs with a restricted input space in the $\mathsf{GGAM}^{\mathsf{T}}$ model, and then prove the security of the expanded PRFs, namely the resulting SMP-PRFs, via security reduction under the GA-DDH assumption. This provides the rigorous security proof for poly-style PRFs that was missing from prior work. Finally, building on SMP-PRFs, we present a new efficient VRF construction.

Nevertheless, the quantum security of SMP-PRFs warrants further investigation. Although security in the $\mathrm{GGAM}^{\mathsf{T}}$ model suffices to rule out efficient generic group-action attacks of the type considered in this work, the model does not capture fully quantum adversaries. In [19], Duman et al. extend this model to the quantum setting by allowing the adversary to query the group-action oracle in superposition. However, deriving and evaluating meaningful bounds for computational problems in this model remains challenging. We leave this problem for future work.

References

1. Abdalla, M., Eisenhofer, T., Kiltz, E., Kunzweiler, S., Riepel, D.: Password-authenticated key exchange from group actions. In: Annual international cryptology conference. pp. 699–728. Springer (2022). https://doi.org/10.1007/978-3-031-15979-4_24
2. Alamati, N., De Feo, L., Montgomery, H., Patranabis, S.: Cryptographic group actions and applications. In: Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part II 26. pp. 411–439. Springer (2020). https://doi.org/10.1007/978-3-030-64834-3_14
3. Badrinarayanan, S., Masny, D., Mukherjee, P., Patranabis, S., Raghuraman, S., Sarkar, P.: Round-optimal oblivious transfer and MPC from computational CSIDH. In: IACR International Conference on Public-Key Cryptography. pp. 376–405. Springer (2023). https://doi.org/10.1007/978-3-031-31368-4_14
4. Bernstein, D.J., Lange, T., Martindale, C., Panny, L.: Quantum circuits for the CSIDH: optimizing quantum evaluation of isogenies. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 409–441. Springer (2019). https://doi.org/10.1007/978-3-030-17656-3_15
5. Beullens, W., Dobson, S., Katsumata, S., Lai, Y.F., Pintore, F.: Group signatures and more from isogenies and lattices: Generic, simple, and efficient. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 95–126. Springer (2022). https://doi.org/10.1007/978-3-031-07085-3_4
6. Beullens, W., Katsumata, S., Pintore, F.: Calamari and falafl: logarithmic (linkable) ring signatures from isogenies and lattices. In: International Conference on

the Theory and Application of Cryptology and Information Security. pp. 464–492. Springer (2020). https://doi.org/10.1007/978-3-030-64834-3_16
7. Beullens, W., Kleinjung, T., Vercauteren, F.: Csi-fish: efficient isogeny based signatures through class group computations. In: International conference on the theory and application of cryptology and information security. pp. 227–247. Springer (2019). https://doi.org/10.1007/978-3-030-34578-5_9
8. Biasse, J.F., Micheli, G., Persichetti, E., Santini, P.: Less is more: code-based signatures without syndromes. In: International Conference on Cryptology in Africa. pp. 45–65. Springer (2020). https://doi.org/10.1007/978-3-030-51938-4_3
9. Boneh, D., Montgomery, H.W., Raghunathan, A.: Algebraic pseudorandom functions with improved efficiency from the augmented cascade. In: Proceedings of the 17th ACM conference on Computer and communications security. pp. 131–140 (2010). https://doi.org/10.1145/1866307.1866323
10. Boneh, D., Shoup, V.: A graduate course in applied cryptography. Draft 0.5 (2020)
11. Bonnetain, X., Schrottenloher, A.: Quantum security analysis of CSIDH. In: Annual international conference on the theory and applications of cryptographic techniques. pp. 493–522. Springer (2020). https://doi.org/10.1007/978-3-030-45724-2_17
12. Borin, G., Persichetti, E., Pintore, F., Reijnders, K., Santini, P.: A guide to the design of digital signatures based on cryptographic group actions. Journal of Cryptology **38**(3), 23 (2025). https://doi.org/10.1007/s00145-025-09542-9
13. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: an efficient post-quantum commutative group action. In: International conference on the theory and application of cryptology and information security. pp. 395–427. Springer (2018). https://doi.org/10.1007/978-3-030-03332-3_15
14. Chen, M., Lai, Y.F., Laval, A., Marco, L., Petit, C.: Malleable commitments from group actions and zero-knowledge proofs for circuits based on isogenies. In: International Conference on Cryptology in India. pp. 221–243. Springer (2023). https://doi.org/10.1007/978-3-031-56232-7_11
15. Dartois, P., Eriksen, J.K., Fouotsa, T.B., Herlédan Le Merdy, A., Invernizzi, R., Robert, D., Rueger, R., Vercauteren, F., Wesolowski, B.: Pegasis: Practical effective class group action using 4-dimensional isogenies. In: Annual International Cryptology Conference. pp. 67–99. Springer (2025). https://doi.org/10.1007/978-3-032-01855-7_3
16. De Feo, L., Fouotsa, T.B., Kutas, P., Leroux, A., Merz, S.P., Panny, L., Wesolowski, B.: Scallop: scaling the csi-fish. In: IACR international conference on public-key cryptography. pp. 345–375. Springer (2023). https://doi.org/10.1007/978-3-031-31368-4_13
17. De Feo, L., Meyer, M.: Threshold schemes from isogeny assumptions. In: IACR International Conference on Public-Key Cryptography. pp. 187–212. Springer (2020). https://doi.org/10.1007/978-3-030-45388-6_7
18. Ducas, L., van Woerden, W.: On the lattice isomorphism problem, quadratic forms, remarkable lattices, and cryptography. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 643–673. Springer (2022). https://doi.org/10.1007/978-3-031-07082-2_23
19. Duman, J., Hartmann, D., Kiltz, E., Kunzweiler, S., Lehmann, J., Riepel, D.: Generic models for group actions. In: IACR International Conference on Public-Key Cryptography. pp. 406–435. Springer (2023). https://doi.org/10.1007/978-3-031-31368-4_15

20. Duong, D.H., Khuc, T.X., Qiao, Y., Susilo, W., Zhang, C.: Blind signatures from cryptographic group actions. IACR Cryptol. ePrint Arch. p. 397 (2025), https://eprint.iacr.org/2025/397
21. 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)
22. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. Journal of the ACM (JACM) **33**(4), 792–807 (1986). https://doi.org/10.1145/6490.6503
23. Haviv, I., Regev, O.: On the lattice isomorphism problem. In: Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms. pp. 391–404. SIAM (2014). https://doi.org/10.1137/1.9781611973402.29
24. Ji, Z., Qiao, Y., Song, F., Yun, A.: General linear group action on tensors: A candidate for post-quantum cryptography. In: Theory of cryptography conference. pp. 251–281. Springer (2019). https://doi.org/10.1007/978-3-030-36030-6_11
25. Jiang, K., Wang, A., Luo, H., Liu, G., Tang, G., Pan, Y., Wang, X.: Re-randomize and extract: A novel commitment construction framework based on group actions. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 124–153. Springer (2025). https://doi.org/10.1007/978-3-031-91124-8_5
26. Jiang, K., Wang, A., Luo, H., Liu, G., Yu, Y., Wang, X.: Exploiting the symmetry of $\mathbb{Z}^n$: Randomization and the automorphism problem. Journal of Cryptology **38**(3), 27 (2025). https://doi.org/10.1007/s00145-025-09546-5
27. 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. Designs, Codes and Cryptography **92**(11), 3587–3643 (2024). https://doi.org/10.1007/s10623-024-01441-7
28. Lai, Y.F.: Capybara and Tsubaki: Verifiable random functions from group actions and isogenies. IACR Communications in Cryptology **1**(3) (2024). https://doi.org/10.62056/avr-11zn4
29. Lai, Y.F.: A note on isogeny group action-based pseudorandom functions. Cryptology ePrint Archive (2024)
30. Lai, Y.F., Galbraith, S.D., Delpech de Saint Guilhem, C.: Compact, efficient and uc-secure isogeny-based oblivious transfer. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 213–241. Springer (2021). https://doi.org/10.1007/978-3-030-77870-5_8
31. Lai, Y.F., Yu, Y., Zhou, X.: Triple cryptanalysis of isogeny-based vrfs from asiacrypt 2025. Cryptology ePrint Archive (2026)
32. Levin, S., Pedersen, R.: Faster proofs and VRFs from isogenies. ASIACRYPT 2025 (2025). https://doi.org/10.1007/978-981-95-5113-2_10
33. Micali, S., Rabin, M., Vadhan, S.: Verifiable random functions. In: 40th annual symposium on foundations of computer science (cat. No. 99CB37039). pp. 120–130. IEEE (1999). https://doi.org/10.1109/SFFCS.1999.814584
34. Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. Journal of the ACM (JACM) **51**(2), 231–262 (2004). https://doi.org/10.1145/972639.972643
35. Naor, M., Reingold, O., Rosen, A.: Pseudo-random functions and factoring. In: Proceedings of the thirty-second annual ACM symposium on Theory of computing. pp. 11–20 (2000). https://doi.org/10.1145/335305.335307
36. Panny, L., Petit, C., Stopar, M.: Klapoti: An asymptotically efficient isogeny group action from 2-dimensional isogenies. Cryptology ePrint Archive (2024)

37. Panny, L., Petit, C., Stopar, M.: Klapoti: An asymptotically efficient isogeny group action from 2-dimensional isogenies. IACR Communications in Cryptology **2**(3) (2025). https://doi.org/10.62056/ahp2wakrz
38. Patarin, J.: Hidden fields equations (hfe) and isomorphisms of polynomials (ip): Two new families of asymmetric algorithms. In: International conference on the theory and applications of cryptographic techniques. pp. 33–48. Springer (1996). https://doi.org/10.1007/3-540-68339-9_4
39. Pedersen, R.: Threshold oblivious pseudorandom functions from isogeny group actions. Cryptology ePrint Archive (2026)
40. Peikert, C.: He gives c-sieves on the CSIDH. In: Annual international conference on the theory and applications of cryptographic techniques. pp. 463–492. Springer (2020). https://doi.org/10.1007/978-3-030-45724-2_16
41. Pham, M.T.T., Duong, D.H., Li, Y., Susilo, W.: Threshold ring signature scheme from cryptographic group action. In: International Conference on Provable Security. pp. 207–227. Springer (2023). https://doi.org/10.1007/978-3-031-45513-1_12
42. Delpech de Saint Guilhem, C., Pedersen, R.: New proof systems and an oprf from CSIDH. In: IACR International Conference on Public-Key Cryptography. pp. 217–251. Springer (2024). https://doi.org/10.1007/978-3-031-57725-3_8
43. Sendrier, N., Simos, D.E.: The hardness of code equivalence over $\mathbb{F}_q$ and its application to code-based cryptography. In: International Workshop on Post-Quantum Cryptography. pp. 203–216. Springer (2013). https://doi.org/10.1007/978-3-642-38616-9_14
44. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM review **41**(2), 303–332 (1999). https://doi.org/10.1137/S0036144598347011
45. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 256–266. Springer (1997). https://doi.org/10.1007/3-540-69053-0_18
46. Tang, G., Duong, D.H., Joux, A., Plantard, T., Qiao, Y., Susilo, W.: Practical post-quantum signature schemes from isomorphism problems of trilinear forms. In: Annual international conference on the theory and applications of cryptographic techniques. pp. 582–612. Springer (2022). https://doi.org/10.1007/978-3-031-07082-2_21

A Proof of Equivalence of Pseudorandomness Definitions

Theorem 17 *The pseudorandomness notions in Definition 4 and Equivalent Definition 1 of Section 2.2 are equivalent.*

Proof. 1. If there exists an algorithm $\mathcal{A}$ achieving advantage ε in Definition 4, then we can construct a PPT algorithm $\mathcal{B}$ such that $\mathcal{B}^{\mathcal{A}}$ achieving the same advantage ε in *Equivalent Definition 1*.

The construction is straightforward. Whenever $\mathcal{A}$ makes its i-th query x_i ($i \in [q]$), $\mathcal{B}$ queries $\mathcal{O}_{b,k}^{\mathsf{PR}'}$ on input x_i and forwards the first component of the response to $\mathcal{A}$. After completing its queries, $\mathcal{A}$ outputs a bit b^*, which $\mathcal{B}$ outputs as its own guess. Since the first component returned by $\mathcal{O}_{b,k}^{\mathsf{PR}'}$ is distributed identically to the response of $\mathcal{O}_{b,k}^{\mathsf{PR}}$, $\mathcal{B}$ perfectly simulates the oracle for $\mathcal{A}$. Consequently, $\mathcal{B}$

wins if and only if $\mathcal{A}$ wins, and hence $\mathcal{B}$ achieves the same advantage as $\mathcal{A}$.

2. If there exists an algorithm $\mathcal{A}$ achieving advantage $\varepsilon_{\mathcal{A}}$ in *Equivalent Definition 1*, then we can construct a PPT algorithm $\mathcal{B}$ such that $\mathcal{B}^{\mathcal{A}}$ achieving advantage $\varepsilon_{\mathcal{B}} = \frac{1}{2}\varepsilon_{\mathcal{A}}$ in Definition 4.

First, $\mathcal{B}$ samples a bit $\delta \xleftarrow{\$} \{0,1\}$ and initializes an empty list $\mathcal{L} \leftarrow \varnothing$. Whenever $\mathcal{A}$ makes its i-th query x_i ($i \in [q]$), $\mathcal{B}$ queries its oracle $\mathcal{O}_{b,k}^{\mathsf{PR}}$ on input x_i and obtains $\mathcal{O}_{b,k}^{\mathsf{PR}}(x_i)$. It then checks whether there exists a pair $(x, r) \in \mathcal{L}$ such that $x_i = x$. If so, $\mathcal{B}$ sets $r_i \leftarrow r$. Otherwise, $\mathcal{B}$ samples $r_i \xleftarrow{\$} \mathcal{Y}_\lambda$ and adds (x_i, r_i) to the list $\mathcal{L}$. If $\delta = 0$, $\mathcal{B}$ returns $(r_i, \mathcal{O}_{b,k}^{\mathsf{PR}}(x_i))$ to $\mathcal{A}$; otherwise, it returns $(\mathcal{O}_{b,k}^{\mathsf{PR}}(x_i), r_i)$. After completing its queries, $\mathcal{A}$ outputs a bit b'. Then, $\mathcal{B}$ sets its guess for b to $b^* \leftarrow 1$ if $b' = \delta$, and to $b^* \leftarrow 0$ otherwise.

When $b = 1$, $\mathcal{B}$ provides a perfect simulation of the oracle $\mathcal{O}_{\delta,k}^{\mathsf{PR}'}$ for $\mathcal{A}$. When $b = 0$, the two components returned to $\mathcal{A}$ are evaluations of two independent random functions from $\mathcal{X}_\lambda$ to $\mathcal{Y}_\lambda$. Therefore, the view of $\mathcal{A}$ is independent of δ, and hence $\Pr[b' = \delta \mid b = 0] = \Pr[b' \neq \delta \mid b = 0] = 1/2$. Thus, it follows that

$$\begin{aligned}\Pr[\mathcal{B} \text{ wins}] &= \frac{1}{2}\Pr[b' \neq \delta \mid b = 0] + \frac{1}{2}\Pr[b' = \delta \mid b = 1] \\ &= \frac{1}{2}\cdot\frac{1}{2} + \frac{1}{2}\left(\frac{1}{2} + \frac{\varepsilon_{\mathcal{A}}}{2}\right) = \frac{1}{2} + \frac{1}{4}\varepsilon_{\mathcal{A}}.\end{aligned}$$

Therefore, the advantage of $\mathcal{B}$ in the game defined in Definition 4 is

$$\varepsilon_{\mathcal{B}} = 2\left(\Pr[\mathcal{B} \text{ wins}] - \frac{1}{2}\right) = \frac{1}{2}\varepsilon_{\mathcal{A}}.$$

□

B Additional Details on the Proof of Theorem 4

In this section, we provide some additional details on the proof of Theorem 4.

Detailed pseudorandomness experiments for generic algorithm $\mathcal{A}$: Below, we describe the detailed interaction between a generic group action algorithm $\mathcal{A}$ and oracles in the pseudorandomness experiment in Theorem 4. We refer to this experiment as game G. As stated in the theorem, we assume that algorithm $\mathcal{A}$ makes q queries to the oracle $\mathcal{O}_{b,\mathbf{k}}^{\mathcal{S}}$ and a total of q' queries to the group action oracle $\mathcal{O}^{\mathsf{GA}}$ and the twist oracle $\mathcal{O}^{\mathsf{Tw}}$.

• Game G: Denote the two polynomial rings $\mathbb{Z}_N[U_1, \ldots, U_q, V_1, \ldots, V_q]$ and $\mathbb{Z}_N[A_0, A_1, \ldots, A_d, R_1, \ldots, R_q]$ by $\mathcal{R}$ and $\mathcal{R}'$, respectively. This game maintains five lists, denoted by $\mathcal{L}_{\mathsf{q}}, \mathcal{L}_F, \mathcal{L}_{F'}, \mathcal{L}_{\mathrm{y}}$, and $\mathcal{L}_\sigma$, where the lists $\mathcal{L}_{F'}$ and $\mathcal{L}_{\mathrm{y}}$ are kept hidden from the algorithm $\mathcal{A}$, whereas other lists are public. Furthermore, let $\mathcal{I}_F$ be the index set of all elements appearing in $\mathcal{L}_F$ at the end of the game. At the beginning of the game, we randomly choose $\mathbf{k} = (a_0, \ldots, a_d) \xleftarrow{\$} \mathbb{Z}_N^{d+1}$,

$(r_1, \ldots, r_q) \xleftarrow{\$} \mathbb{Z}_N^q$ and $b \xleftarrow{\$} \{0,1\}$, and initialize the five lists as $\varnothing$, $\{F_0\ (=0)\}$ $(\in \mathcal{R})$, $\{F'_0\ (=0)\}$ $(\in \mathcal{R}')$, $\{\mathrm{y}_0\ (=0)\}$ $(\in \mathbb{Z}_N)$, and $\{\sigma_0\ (\xleftarrow{\$} \Sigma)\}$, respectively. At any step in the game, the algorithm $\mathcal{A}$ may issue a query $\mathbf{Q}_k$ (assume that this is the k-th query $(k \in [q+q'])$), which falls into one of the following three types. We first record $\mathbf{Q}_k$ in the list $\mathcal{L}_{\mathsf{q}}$. Then, depending on the type of the query, we append new values to the lists $\mathcal{L}_F$, $\mathcal{L}_{F'}$ and $\mathcal{L}_{\mathrm{y}}$ as described below.

- If $\mathbf{Q}_k = (1, \mathsf{ind}, c)$, where ind denotes the index of the polynomial already stored in the list $\mathcal{L}_F$ and $c \in \mathbb{Z}_N$, we compute $F_k = F_{\mathsf{ind}} + c$, $F'_k = F'_{\mathsf{ind}} + c$, $\mathrm{y}_k = F'_k(a_0, \ldots, a_d, r_1, \ldots, r_q)$, then add F_k, F'_k, y_k to the list $\mathcal{L}_F, \mathcal{L}_{F'}, \mathcal{L}_{\mathrm{y}}$, respectively.
- If $\mathbf{Q}_k = (2, \mathsf{ind})$ where ind denotes the index of the polynomial already stored in the list $\mathcal{L}_F$, we compute $F_k = -F_{\mathsf{ind}},\ F'_k = -F'_{\mathsf{ind}},\ \mathrm{y}_k = F'_k(a_0, \ldots, a_d, r_1, \ldots, r_q)$, then add F_k, F'_k, y_k to lists $\mathcal{L}_F, \mathcal{L}_{F'}, \mathcal{L}_{\mathrm{y}}$, respectively.
- If $\mathbf{Q}_k = (3, \mathrm{x}_l)$, where $l \in [q]$ denotes that this corresponds to the l-th query[20] of the third type made by $\mathcal{A}$, we set $F_{k,0} = U_l,\ F_{k,1} = V_l$. If $b = 0$, we set $F'_{k,0} = R_l$ and $F'_{k,1} = \mathrm{x}_l^d A_d + \cdots + \mathrm{x}_l A_1 + A_0$; if $b = 1$, we set $F'_{k,0} = \mathrm{x}_l^d A_d + \cdots + \mathrm{x}_l A_1 + A_0$ and $F'_{k,1} = R_l$. Then we compute $\mathrm{y}_{k,\delta} = F'_{k,\delta}(a_0, \ldots, a_d, r_1, \ldots, r_q)$ for $\delta \in \{0,1\}$, and append $(F_{k,0}, F_{k,1})$, $(F'_{k,0}, F'_{k,1})$ and $(\mathrm{y}_{k,0}, \mathrm{y}_{k,1})$ to lists $\mathcal{L}_F, \mathcal{L}_{F'}, \mathcal{L}_{\mathrm{y}}$, respectively.

Finally, we add the label $\sigma_k \in \Sigma$ (or the tuple of labels $(\sigma_{k,0}, \sigma_{k,1}) \in \Sigma^2$, depending on the type of query issued by $\mathcal{A}$) to the list $\mathcal{L}_\sigma$. The labels are assigned according to the following rule. For the newly computed value y_k (or, in the case of the third type of query, for each component $\mathrm{y}_{k,\delta}$ with $\delta \in \{0,1\}$), we check whether it collides with any previously recorded value already recorded in $\mathcal{L}_{\mathrm{y}}$. If such a collision occurs, we assign the same label as the corresponding previously recorded value; otherwise, we sample a fresh label uniformly at random from the set of labels that have not been used before.

The algorithm $\mathcal{A}$ may perform arbitrary internal computations and eventually output b^* as its guess for b. We say that $\mathcal{A}$ wins the game if $b^* = b$.

We define $\mathsf{Coll}_{\mathsf{G}}$ to be the event that there exists $i, j \in \mathcal{I}_F$ with $i \neq j$ such that $\mathrm{y}_i = \mathrm{y}_j \ \wedge\ F_i \neq F_j$ (Note that an index may be either of the form $k \in [0, q+q']$ or of the form $(k, \delta) \in [0, q+q'] \times \{0,1\}$. Two indices are considered equal only if both their forms and their values coincide).

Remark: In this game, we can assume that the polynomials recorded in $\mathcal{L}_F$ are pairwise distinct for simplicity. This assumption is justified because $\mathcal{L}_F$ is public to $\mathcal{A}$, and generating a polynomial that already appears in $\mathcal{L}_F$ provides no new information to the adversary. Hence, such redundant queries can be omitted without affecting the view or the advantage of $\mathcal{A}$.

[20] To simplify our notation, we assume that type-3 queries issued by $\mathcal{A}$ are made on pairwise distinct input $x_1, \ldots, x_q$, and $x_l \in \mathcal{S}$ $(\forall\ l \in [q])$. For repeated queries, it suffices to return exactly the same polynomials in $\mathcal{L}_F$. If $\mathrm{x}_l \notin \mathcal{S}$, we do nothing and simply wait for the next query issued by $\mathcal{A}$.

Some Details on Upper-Bounding $\Pr[\mathsf{Coll}]$*:* One can observe that, strictly speaking, the collision probability upper-bounded in the proof of Theorem 4 in the main body is slightly different from $\Pr[\mathsf{Coll}_{\mathsf{G}}]$ as defined in Game G. In the proof in the main body, we conceptually assume that the adversary has already issued all of its queries, and only afterwards substitute the sampled key $\mathbf{k}$ and the randomness $\mathbf{r}$ of the truly random function into the resulting polynomials to check whether any collision occurs. In the actual Game G, however, the corresponding polynomial values are evaluated during the interaction, and any collision that occurs may affect the labels returned to the adversary and hence its subsequent queries. To bridge this gap, we introduce the following Game G'.

• Game G': At the beginning of the game, this game is largely the same as Game G with respect to the sampling and initialization procedures, except that this game does not require sampling the bit b. Subsequently, whenever $\mathcal{A}$ issues a query, we append the corresponding query and polynomials to the lists $\mathcal{L}_{\mathsf{q}}$ and $\mathcal{L}_F$, respectively, in exactly the same manner as in Game G. Next, we skip the updates to the lists $\mathcal{L}_{F'}$ and $\mathcal{L}_{\mathrm{y}}$ (note that these two lists are originally hidden from $\mathcal{A}$ in game G) and instead add labels directly to the list $\mathcal{L}_\sigma$. Unlike in Game G, the labels are assigned according to the list $\mathcal{L}_F$, rather than the list $\mathcal{L}_{\mathrm{y}}$. Specifically, whenever a new polynomial is added to $\mathcal{L}_F$, we check whether the same polynomial has already appeared in the list. If so, we assign it the same label as the previously recorded copy; otherwise, we sample a fresh label uniformly at random from the unused labels and add it to $\mathcal{L}_\sigma$. We define an event $\mathsf{Coll}_{\mathsf{G}'}$ to be the event that at least one of the following two events occurs:

- Let $U_l = r_l,\ V_l = a_0 + \mathrm{x}_l a_1 + \cdots + \mathrm{x}_l^d a_d$ for $l \in [q]$, compute $\mathrm{y}_{\mathsf{ind}}^{(0)} = F_{\mathsf{ind}}(U_1, \ldots, U_q, V_1, \ldots, V_q)$ for $\mathsf{ind} \in \mathcal{I}_F$, and let $\mathcal{L}_{\mathrm{y}}^{(0)} = \{\mathrm{y}_{\mathsf{ind}}^{(0)}\}_{\mathsf{ind}\in\mathcal{I}_F}$. Then, there exists $i, j \in \mathcal{I}_F$ with $i \neq j$ such that $\mathrm{y}_i^{(0)} = \mathrm{y}_j^{(0)} \ \wedge\ F_i \neq F_j$.
- Let $U_l = a_0 + \mathrm{x}_l a_1 + \cdots + \mathrm{x}_l^d a_d,\ V_l = r_l$ for $l \in [q]$, compute $\mathrm{y}_{\mathsf{ind}}^{(1)} = F_{\mathsf{ind}}(U_1, \ldots, U_q, V_1, \ldots, V_q)$ for $\mathsf{ind} \in \mathcal{I}_F$, and let $\mathcal{L}_{\mathrm{y}}^{(1)} = \{\mathrm{y}_{\mathsf{ind}}^{(1)}\}_{\mathsf{ind}\in\mathcal{I}_F}$. Then, there exists $i, j \in \mathcal{I}_F$ with $i \neq j$ such that $\mathrm{y}_i^{(1)} = \mathrm{y}_j^{(1)} \ \wedge\ F_i \neq F_j$.

In the main body, what we actually upper-bound is the probability of the event $\mathsf{Coll}_{\mathsf{G}'}$ defined in Game G' above. In fact, we have

$$\Pr[\mathsf{Coll}_{\mathsf{G}}] \leqslant \Pr[\mathsf{Coll}_{\mathsf{G}'}].$$

To see this, fix $\delta \in \{0, 1\}$ and consider Game G conditioned on $b = \delta$. Before the first collision under the substitution corresponding to $b = \delta$ occurs, distinct polynomials in $\mathcal{L}_F$ evaluate to distinct values, and hence Game G assigns labels in exactly the same manner as Game G'. Consequently, $\mathcal{A}$ observes the same transcript and issues the same queries in the two games up to the first such collision. Therefore, whenever $\mathsf{Coll}_{\mathsf{G}}$ occurs with $b = \delta$, the corresponding collision under the substitution for $b = \delta$ is also detected by $\mathsf{Coll}_{\mathsf{G}'}$. Since $\mathsf{Coll}_{\mathsf{G}'}$ checks

both $\delta = 0$ and $\delta = 1$, the above inequality follows. This technique already appears in the work on the generic group model [45].

Detailed Derivation of the Upper Bound on $\Pr[\mathcal{A}$ *wins* $\mid \overline{\mathsf{Coll}}]$[21]: Here, we introduce three random variables. The first random variable, denoted by $\boldsymbol{\xi}$, represents the random values chosen at the beginning of the experiment, namely, $(\mathbf{k}, \mathbf{r}) = (a_0, \ldots, a_d, r_1, \ldots, r_q)$. Clearly, $\boldsymbol{\xi}$ is uniformly distributed over $\mathbb{Z}_N^{d+q+1}$. The second random variable, denoted by $\boldsymbol{\zeta}$, represents the vector consisting of all queries (recorded in $\mathcal{L}_{\mathsf{q}}$) issued by $\mathcal{A}$ throughout the game G, namely, $\mathbf{Q} = (\mathbf{Q}_1, \ldots, \mathbf{Q}_{q+q'})$. The third random variable, denoted by $\boldsymbol{\eta}$, represents the vector consisting of all labels (recorded in $\mathcal{L}_\sigma$) sampled during the game G, namely, $\boldsymbol{\sigma} = (\boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_{q+q'})$, where $\boldsymbol{\sigma}_i = \sigma_i$ or $(\sigma_{i,0}, \sigma_{i,1})$, depending on the type of query. Note that the random variable $\boldsymbol{\sigma}$ does not include the initial label $\boldsymbol{\sigma}_0 = \sigma_E(E_0)$, which is sampled at the beginning of the experiment and given to $\mathcal{A}$ as part of its initial input. Throughout the following analysis, we condition on σ_0 taking an arbitrary fixed value. We omit this conditioning from the notation for simplicity.

Our goal is to upper-bound the probability $\Pr[\mathcal{A}$ *wins* $\mid \overline{\mathsf{Coll}}]$. Note that, conditioned on the event $\overline{\mathsf{Coll}}$, the random variables $\boldsymbol{\zeta}$ and $\boldsymbol{\eta}$ can only take certain values (for example, the label components of $\boldsymbol{\eta}$ corresponding to distinct polynomials in $\mathcal{L}_F$ must be pairwise distinct). We define

$$\mathsf{Supp} := \left\{ (\mathbf{Q}, \boldsymbol{\sigma}) \;\middle|\; \Pr[\boldsymbol{\zeta} = \mathbf{Q}, \boldsymbol{\eta} = \boldsymbol{\sigma} \,|\, \overline{\mathsf{Coll}}] > 0 \right\}.$$

Then, the probability we need to upper-bound can be written as

$$\begin{aligned} &\Pr[\mathcal{A} \textit{ wins} \mid \overline{\mathsf{Coll}}] \\ &= \sum_{(\mathbf{Q},\boldsymbol{\sigma}) \in \mathsf{Supp}} \Pr[\mathcal{A} \textit{ wins} \mid \boldsymbol{\zeta} = \mathbf{Q}, \boldsymbol{\eta} = \boldsymbol{\sigma}, \overline{\mathsf{Coll}}] \Pr[\boldsymbol{\zeta} = \mathbf{Q}, \boldsymbol{\eta} = \boldsymbol{\sigma} \mid \overline{\mathsf{Coll}}]. \end{aligned} \tag{22}$$

Our next goal is to establish a uniform upper bound on $\Pr[\mathcal{A}$ *wins* $\mid \boldsymbol{\zeta} = \mathbf{Q}, \boldsymbol{\eta} = \boldsymbol{\sigma}, \overline{\mathsf{Coll}}]$ for every fixed $(\mathbf{Q}, \boldsymbol{\sigma}) \in \mathsf{Supp}$. Note that for every $(\mathbf{Q}, \boldsymbol{\sigma}) \in \mathsf{Supp}$, the components of $\boldsymbol{\sigma}$ are pairwise distinct and are also distinct from the fixed initial label σ_0, provided that their corresponding polynomials in $\mathcal{L}_F$ are all distinct. Thus, the event $\boldsymbol{\eta} = \boldsymbol{\sigma}$ implies $\overline{\mathsf{Coll}}$. Therefore, it follows that

$$\begin{aligned} &\Pr[\mathcal{A} \textit{ wins} \mid \boldsymbol{\zeta} = \mathbf{Q}, \boldsymbol{\eta} = \boldsymbol{\sigma}, \overline{\mathsf{Coll}}] = \Pr[\mathcal{A} \textit{ wins} \mid \boldsymbol{\zeta} = \mathbf{Q}, \boldsymbol{\eta} = \boldsymbol{\sigma}] \\ &= \sum_{\delta \in \{0,1\}} \Pr[b^* = \delta \mid b = \delta,\ \boldsymbol{\zeta} = \mathbf{Q}, \boldsymbol{\eta} = \boldsymbol{\sigma}] \Pr[b = \delta \mid \boldsymbol{\zeta} = \mathbf{Q},\ \boldsymbol{\eta} = \boldsymbol{\sigma}]. \end{aligned}$$

Next, we prove that, for any $\delta^*, \delta \in \{0, 1\}$,

$$\Pr[b^* = \delta^* \mid b = \delta,\ \boldsymbol{\zeta} = \mathbf{Q},\ \boldsymbol{\eta} = \boldsymbol{\sigma}] = \Pr[b^* = \delta^* \mid \boldsymbol{\zeta} = \mathbf{Q},\ \boldsymbol{\eta} = \boldsymbol{\sigma}]. \tag{23}$$

[21] The event $\overline{\mathsf{Coll}}$ considered in the main body corresponds to $\overline{\mathsf{Coll}_{\mathsf{G}}}$ here. For consistency, we continue to use the notation $\overline{\mathsf{Coll}}$ throughout the following analysis.

Let $\boldsymbol{\Omega}$ denote the random variable corresponding to the random tape of $\mathcal{A}$, and let $\boldsymbol{\omega}$ denote a possible value of $\boldsymbol{\Omega}$. Let $\mathbf{T} = (\boldsymbol{\zeta}, \boldsymbol{\eta})$ denote the transcript, and let $\mathbf{t} = (\mathbf{Q}, \boldsymbol{\sigma})$ be a possible value of $\mathbf{T}$. Define

$$\mathcal{R}_{\mathbf{t}} := \{\boldsymbol{\omega} \mid \mathcal{A} \text{ follows transcript } \mathbf{t} \text{ under random tape } \boldsymbol{\omega}\},$$
$$\mathcal{R}_{\mathbf{t},\delta^*} := \{\boldsymbol{\omega} \mid \mathcal{A} \text{ follows transcript } \mathbf{t} \text{ and outputs } \delta^* \text{ under random tape } \boldsymbol{\omega}\}.$$

Clearly, $\mathcal{R}_{\mathbf{t},\delta^*} \subseteq \mathcal{R}_{\mathbf{t}}$. Then, we have

$$\Pr[b^* = \delta^* \mid \mathbf{T} = \mathbf{t}, b = \delta] = \sum_{\boldsymbol{\omega} \in \mathcal{R}_{\mathbf{t},\delta^*}} \Pr[\boldsymbol{\Omega} = \boldsymbol{\omega} \mid \mathbf{T} = \mathbf{t}, b = \delta].$$

By Bayes' rule, for any $\boldsymbol{\omega} \in \mathcal{R}_{\mathbf{t},\delta^*}$, it follows that

$$\begin{aligned}\Pr[\boldsymbol{\Omega} = \boldsymbol{\omega} \mid \mathbf{T} = \mathbf{t}, b = \delta] &= \frac{\Pr[\mathbf{T} = \mathbf{t} \mid \boldsymbol{\Omega} = \boldsymbol{\omega},\ b = \delta] \Pr[\boldsymbol{\Omega} = \boldsymbol{\omega} \mid b = \delta]}{\Pr[\mathbf{T} = \mathbf{t} \mid b = \delta]} \\ &= \frac{\Pr[\mathbf{T} = \mathbf{t} \mid \boldsymbol{\Omega} = \boldsymbol{\omega}, b = \delta] \Pr[\boldsymbol{\Omega} = \boldsymbol{\omega} \mid b = \delta]}{\sum_{\boldsymbol{\omega}' \in \mathcal{R}_{\mathbf{t}}} \Pr[\mathbf{T} = \mathbf{t} \mid \boldsymbol{\Omega} = \boldsymbol{\omega}', b = \delta] \Pr[\boldsymbol{\Omega} = \boldsymbol{\omega}' \mid b = \delta]}\end{aligned}$$

Note that, for any $\boldsymbol{\omega}' \in \mathcal{R}_{\mathbf{t}}$, the probability $\Pr[\mathbf{T} = \mathbf{t} \mid \boldsymbol{\Omega} = \boldsymbol{\omega}',\ b = \delta]$ takes the same value. This is because the challenger has no access to the random tape $\boldsymbol{\omega}'$ of $\mathcal{A}$ and only observes the query sequence specified by $\mathbf{t}$.

Moreover, since $\boldsymbol{\Omega}$ and b are sampled independently at the beginning of the experiment, we have

$$\Pr[\boldsymbol{\Omega} = \boldsymbol{\omega} \mid b = \delta] = \Pr[\boldsymbol{\Omega} = \boldsymbol{\omega}]$$

for any $\delta \in \{0, 1\}$. Thus, the above expression simplifies to

$$\Pr[\boldsymbol{\Omega} = \boldsymbol{\omega} \mid \mathbf{T} = \mathbf{t}, b = \delta] = \frac{\Pr[\boldsymbol{\Omega} = \boldsymbol{\omega}]}{\sum_{\boldsymbol{\omega}' \in \mathcal{R}_{\mathbf{t}}} \Pr[\boldsymbol{\Omega} = \boldsymbol{\omega}']}.$$

Therefore,

$$\Pr[b^* = \delta^* \mid \mathbf{T} = \mathbf{t}, b = \delta] = \frac{\sum_{\boldsymbol{\omega} \in \mathcal{R}_{\mathbf{t},\delta^*}} \Pr[\boldsymbol{\Omega} = \boldsymbol{\omega}]}{\sum_{\boldsymbol{\omega}' \in \mathcal{R}_{\mathbf{t}}} \Pr[\boldsymbol{\Omega} = \boldsymbol{\omega}']},$$

which is independent of δ and proves Equation (23). Using Equation (23), we have

$$\begin{aligned}&\Pr[\mathcal{A} \textit{ wins} \mid \boldsymbol{\zeta} = \mathbf{Q}, \boldsymbol{\eta} = \boldsymbol{\sigma}] \\ =\ &\Pr[b^* = 0 \mid \boldsymbol{\zeta} = \mathbf{Q}, \boldsymbol{\eta} = \boldsymbol{\sigma}] \Pr[b = 0 \mid \boldsymbol{\zeta} = \mathbf{Q},\ \boldsymbol{\eta} = \boldsymbol{\sigma}] \\ &+ (1 - \Pr[b^* = 0 \mid \boldsymbol{\zeta} = \mathbf{Q}, \boldsymbol{\eta} = \boldsymbol{\sigma}]) \Pr[b = 1 \mid \boldsymbol{\zeta} = \mathbf{Q},\ \boldsymbol{\eta} = \boldsymbol{\sigma}] \\ \leqslant\ &\max\{\Pr[b = 0 \mid \boldsymbol{\zeta} = \mathbf{Q},\ \boldsymbol{\eta} = \boldsymbol{\sigma}],\ \Pr[b = 1 \mid \boldsymbol{\zeta} = \mathbf{Q},\ \boldsymbol{\eta} = \boldsymbol{\sigma}]\}\end{aligned}$$

For the fixed transcript $\mathbf{t} = (\mathbf{Q}, \boldsymbol{\sigma}) \in \mathsf{Supp}$, let $\Psi_{\delta,\mathbf{t}}$ ($\delta \in \{0,1\}$) denote the set Ψ_δ uniquely determined by $\mathbf{t}$ (see proof of Theorem 4 for the definition of the set Ψ_δ). Since any $(\mathbf{k}, \mathbf{r}) \in \Psi_{\delta,\mathbf{t}}$ induces a collision and hence cannot produce the collision-free transcript $\mathbf{t}$, we have

$$\Pr[b = \delta \mid \boldsymbol{\zeta} = \mathbf{Q},\ \boldsymbol{\eta} = \boldsymbol{\sigma}] = \sum_{(\mathbf{k},\mathbf{r}) \notin \Psi_{\delta,\mathbf{t}}} \Pr[b = \delta,\ \boldsymbol{\xi} = (\mathbf{k}, \mathbf{r}) | \ \boldsymbol{\zeta} = \mathbf{Q},\ \boldsymbol{\eta} = \boldsymbol{\sigma}]. \quad (24)$$

Our next step is to prove that, for any $(\delta, \mathbf{k}, \mathbf{r})$ and $(\delta', \mathbf{k}', \mathbf{r}')$ such that $(\mathbf{k}, \mathbf{r}) \notin \Psi_{\delta,\mathbf{t}}$ and $(\mathbf{k}', \mathbf{r}') \notin \Psi_{\delta',\mathbf{t}}$, respectively, the following equality holds:

$$\begin{aligned} &\Pr[b = \delta,\ \boldsymbol{\xi} = (\mathbf{k}, \mathbf{r}) \mid \boldsymbol{\zeta} = \mathbf{Q},\ \boldsymbol{\eta} = \boldsymbol{\sigma}] \\ =&\Pr[b = \delta',\ \boldsymbol{\xi} = (\mathbf{k}', \mathbf{r}') \mid \boldsymbol{\zeta} = \mathbf{Q},\ \boldsymbol{\eta} = \boldsymbol{\sigma}]. \end{aligned} \quad (25)$$

Given that the above equality holds, the value of $\Pr[b = 0 \mid \boldsymbol{\zeta} = \mathbf{Q},\ \boldsymbol{\eta} = \boldsymbol{\sigma}]$ and $\Pr[b = 1 \mid \boldsymbol{\zeta} = \mathbf{Q},\ \boldsymbol{\eta} = \boldsymbol{\sigma}]$ can be directly derived from the cardinalities of the sets $\mathbb{Z}_N^{d+q+1} \setminus \Psi_{0,\mathbf{t}}$ and $\mathbb{Z}_N^{d+q+1} \setminus \Psi_{1,\mathbf{t}}$.

We now prove that Equation (25) holds.

By Bayes' theorem, it holds that

$$\begin{aligned} &\Pr[b = \delta,\ \boldsymbol{\xi} = (\mathbf{k}, \mathbf{r}) \mid \boldsymbol{\zeta} = \mathbf{Q},\ \boldsymbol{\eta} = \boldsymbol{\sigma}] \\ =&\Pr[\boldsymbol{\zeta} = \mathbf{Q},\ \boldsymbol{\eta} = \boldsymbol{\sigma} \mid b = \delta,\ \boldsymbol{\xi} = (\mathbf{k}, \mathbf{r})] \cdot \frac{\Pr[b = \delta,\ \boldsymbol{\xi} = (\mathbf{k}, \mathbf{r})]}{\Pr[\boldsymbol{\zeta} = \mathbf{Q},\ \boldsymbol{\eta} = \boldsymbol{\sigma}]} \\ &\Pr[b = \delta',\ \boldsymbol{\xi} = (\mathbf{k}', \mathbf{r}') \mid \boldsymbol{\zeta} = \mathbf{Q},\ \boldsymbol{\eta} = \boldsymbol{\sigma}] \\ =&\Pr[\boldsymbol{\zeta} = \mathbf{Q},\ \boldsymbol{\eta} = \boldsymbol{\sigma} \mid b = \delta',\ \boldsymbol{\xi} = (\mathbf{k}', \mathbf{r}')] \cdot \frac{\Pr[b = \delta',\ \boldsymbol{\xi} = (\mathbf{k}', \mathbf{r}')]}{\Pr[\boldsymbol{\zeta} = \mathbf{Q},\ \boldsymbol{\eta} = \boldsymbol{\sigma}]}. \end{aligned} \quad (26)$$

Note that

$$\Pr[b = \delta,\ \boldsymbol{\xi} = (\mathbf{k}, \mathbf{r})] = \Pr[b = \delta',\ \boldsymbol{\xi} = (\mathbf{k}', \mathbf{r}')],$$

therefore, it suffices to prove

$$\Pr[\boldsymbol{\zeta} = \mathbf{Q},\ \boldsymbol{\eta} = \boldsymbol{\sigma} \mid b = \delta,\ \boldsymbol{\xi} = (\mathbf{k}, \mathbf{r})] = \Pr[\boldsymbol{\zeta} = \mathbf{Q},\ \boldsymbol{\eta} = \boldsymbol{\sigma} \mid b = \delta',\ \boldsymbol{\xi} = (\mathbf{k}', \mathbf{r}')]. \quad (27)$$

Next, we apply the chain rule of probability to decompose the joint probability into a product of conditional probabilities, and show that each corresponding

term is identical. Specifically, it holds that

$$
\begin{aligned}
&\Pr[\boldsymbol{\zeta}=\mathbf{Q},\ \boldsymbol{\eta}=\boldsymbol{\sigma}\mid b=\delta,\ \boldsymbol{\xi}=(\mathbf{k},\mathbf{r})]\\
=&\Pr[\boldsymbol{\zeta}[1]=\mathbf{Q}_1\mid b=\delta,\ \boldsymbol{\xi}=(\mathbf{k},\mathbf{r})]\Pr[\boldsymbol{\eta}[1]=\boldsymbol{\sigma}_1\mid\boldsymbol{\zeta}[1]=\mathbf{Q}_1,\ b=\delta,\ \boldsymbol{\xi}=(\mathbf{k},\mathbf{r})]\\
&\cdot\prod_{i=2}^{q+q'}\left(\Pr\left[\boldsymbol{\zeta}[i]=\mathbf{Q}_i\ \middle|\ \begin{array}{c}\boldsymbol{\zeta}[1,i-1]=(\mathbf{Q}_1,\ldots,\mathbf{Q}_{i-1}),\\ \boldsymbol{\eta}[1,i-1]=(\boldsymbol{\sigma}_1,\ldots,\boldsymbol{\sigma}_{i-1}),\\ b=\delta,\ \boldsymbol{\xi}=(\mathbf{k},\mathbf{r})\end{array}\right]\right.\\
&\qquad\left.\cdot\Pr\left[\boldsymbol{\eta}[i]=\boldsymbol{\sigma}_i\ \middle|\ \begin{array}{c}\boldsymbol{\zeta}[1,i]=(\mathbf{Q}_1,\ldots,\mathbf{Q}_i),\\ \boldsymbol{\eta}[1,i-1]=(\boldsymbol{\sigma}_1,\ldots,\boldsymbol{\sigma}_{i-1}),\\ b=\delta,\ \boldsymbol{\xi}=(\mathbf{k},\mathbf{r})\end{array}\right]\right),
\end{aligned}
$$

and

$$
\begin{aligned}
&\Pr[\boldsymbol{\zeta}=\mathbf{Q},\ \boldsymbol{\eta}=\boldsymbol{\sigma}\mid b=\delta',\ \boldsymbol{\xi}=(\mathbf{k}',\mathbf{r}')]\\
=&\Pr[\boldsymbol{\zeta}[1]=\mathbf{Q}_1\mid b=\delta',\ \boldsymbol{\xi}=(\mathbf{k}',\mathbf{r}')]\Pr[\boldsymbol{\eta}[1]=\boldsymbol{\sigma}_1\mid\boldsymbol{\zeta}[1]=\mathbf{Q}_1,\ b=\delta',\ \boldsymbol{\xi}=(\mathbf{k}',\mathbf{r}')]\\
&\cdot\prod_{i=2}^{q+q'}\left(\Pr\left[\boldsymbol{\zeta}[i]=\mathbf{Q}_i\ \middle|\ \begin{array}{c}\boldsymbol{\zeta}[1,i-1]=(\mathbf{Q}_1,\ldots,\mathbf{Q}_{i-1}),\\ \boldsymbol{\eta}[1,i-1]=(\boldsymbol{\sigma}_1,\ldots,\boldsymbol{\sigma}_{i-1}),\\ b=\delta',\ \boldsymbol{\xi}=(\mathbf{k}',\mathbf{r}')\end{array}\right]\right.\\
&\qquad\left.\cdot\Pr\left[\boldsymbol{\eta}[i]=\boldsymbol{\sigma}_i\ \middle|\ \begin{array}{c}\boldsymbol{\zeta}[1,i]=(\mathbf{Q}_1,\ldots,\mathbf{Q}_i),\\ \boldsymbol{\eta}[1,i-1]=(\boldsymbol{\sigma}_1,\ldots,\boldsymbol{\sigma}_{i-1}),\\ b=\delta',\ \boldsymbol{\xi}=(\mathbf{k}',\mathbf{r}')\end{array}\right]\right).
\end{aligned}
\tag{28}
$$

Note that

$$
\Pr[\boldsymbol{\zeta}[1]=\mathbf{Q}_1\mid b=\delta,\ \boldsymbol{\xi}=(\mathbf{k},\mathbf{r})]=\Pr[\boldsymbol{\zeta}[1]=\mathbf{Q}_1\mid b=\delta',\ \boldsymbol{\xi}=(\mathbf{k}',\mathbf{r}')],
\tag{29}
$$

$$
\Pr\left[\boldsymbol{\eta}[1]=\boldsymbol{\sigma}_1\ \middle|\ \begin{array}{c}\boldsymbol{\zeta}[1]=\mathbf{Q}_1,\\ b=\delta,\ \boldsymbol{\xi}=(\mathbf{k},\mathbf{r})\end{array}\right]=\Pr\left[\boldsymbol{\eta}[1]=\boldsymbol{\sigma}_1\ \middle|\ \begin{array}{c}\boldsymbol{\zeta}[1]=\mathbf{Q}_1,\\ b=\delta',\ \boldsymbol{\xi}=(\mathbf{k}',\mathbf{r}')\end{array}\right],
\tag{30}
$$

and

$$
\begin{aligned}
&\Pr\left[\boldsymbol{\eta}[i]=\boldsymbol{\sigma}_i\ \middle|\ \begin{array}{c}\boldsymbol{\zeta}[1,i]=(\mathbf{Q}_1,\ldots,\mathbf{Q}_i),\\ \boldsymbol{\eta}[1,i-1]=(\boldsymbol{\sigma}_1,\ldots,\boldsymbol{\sigma}_{i-1}),\\ b=\delta,\ \boldsymbol{\xi}=(\mathbf{k},\mathbf{r})\end{array}\right]\\
=&\Pr\left[\boldsymbol{\eta}[i]=\boldsymbol{\sigma}_i\ \middle|\ \begin{array}{c}\boldsymbol{\zeta}[1,i]=(\mathbf{Q}_1,\ldots,\mathbf{Q}_i),\\ \boldsymbol{\eta}[1,i-1]=(\boldsymbol{\sigma}_1,\ldots,\boldsymbol{\sigma}_{i-1}),\\ b=\delta',\ \boldsymbol{\xi}=(\mathbf{k}',\mathbf{r}')\end{array}\right].
\end{aligned}
\tag{31}
$$

Moreover, by an argument analogous to that used in deriving Equation (23), we have the following equality:

$$
\begin{aligned}
&\Pr\left[\boldsymbol{\zeta}[i]=\mathbf{Q}_i\ \middle|\ \begin{array}{c}\boldsymbol{\zeta}[1,i-1]=(\mathbf{Q}_1,\ldots,\mathbf{Q}_{i-1}),\\ \boldsymbol{\eta}[1,i-1]=(\boldsymbol{\sigma}_1,\ldots,\boldsymbol{\sigma}_{i-1}),\\ b=\delta,\ \boldsymbol{\xi}=(\mathbf{k},\mathbf{r})\end{array}\right]\\
=&\Pr\left[\boldsymbol{\zeta}[i]=\mathbf{Q}_i\ \middle|\ \begin{array}{c}\boldsymbol{\zeta}[1,i-1]=(\mathbf{Q}_1,\ldots,\mathbf{Q}_{i-1}),\\ \boldsymbol{\eta}[1,i-1]=(\boldsymbol{\sigma}_1,\ldots,\boldsymbol{\sigma}_{i-1}),\\ b=\delta',\ \boldsymbol{\xi}=(\mathbf{k}',\mathbf{r}')\end{array}\right].
\end{aligned}
\tag{32}
$$

Combining Equations (28–32), we immediately obtain that Equation (27) holds, and therefore Equation (25) also holds. Combining Equations (24) and (25), we obtain

$$\begin{aligned}
&\Pr[b=\delta \mid \boldsymbol{\zeta}=\mathbf{Q},\ \boldsymbol{\eta}=\boldsymbol{\sigma}] = \Pr[b=\delta \mid \mathbf{T}=\mathbf{t}] \\
=&\frac{\Pr[b=\delta \mid \boldsymbol{\zeta}=\mathbf{Q},\ \boldsymbol{\eta}=\boldsymbol{\sigma}]}{\Pr[b=0 \mid \boldsymbol{\zeta}=\mathbf{Q},\ \boldsymbol{\eta}=\boldsymbol{\sigma}] + \Pr[b=1 \mid \boldsymbol{\zeta}=\mathbf{Q},\ \boldsymbol{\eta}=\boldsymbol{\sigma}]} \\
=&\frac{|\mathbb{Z}_N^{d+q+1} \setminus \Psi_{\delta,\mathbf{t}}|}{|\mathbb{Z}_N^{d+q+1} \setminus \Psi_{0,\mathbf{t}}| + |\mathbb{Z}_N^{d+q+1} \setminus \Psi_{1,\mathbf{t}}|}
\end{aligned}$$

and thus

$$\max_{\delta\in\{0,1\}} \Pr[b=\delta \mid \mathbf{T}=\mathbf{t}] = \frac{\max\limits_{\delta\in\{0,1\}} \{N^{d+q+1} - |\Psi_{\delta,\mathbf{t}}|\}}{2N^{d+q+1} - |\Psi_{0,\mathbf{t}}| - |\Psi_{1,\mathbf{t}}|}.$$

As for $|\Psi_{\delta,\mathbf{t}}|$, our proof in the main body has already established the upper bound $\binom{2q+q'+1}{2} N^{d+q} D_{\mathcal{S}}$. The above provides a detailed derivation of Inequality (10) stated in the main text.

C Formula for QR(n)

Theorem 18 *(1). Given a positive integer n with prime factorization $n = p_1^{\alpha_1} \cdots p_s^{\alpha_s}$, it holds that*

$$\mathrm{QR}(n) = \prod_{i=1}^{s} \mathrm{QR}(p_i^{\alpha_i}).$$

(2). For an odd prime p and an positive integer α, it holds that

$$\mathrm{QR}(p^{\alpha}) = 1 + \frac{1}{2} \sum_{i=0}^{\lceil \alpha/2 \rceil - 1} \varphi(p^{\alpha-2i}).$$

where φ is the Euler totient function.
(3). For the prime 2 and an positive integer α, it holds that

$$\mathrm{QR}(2^{\alpha}) = \begin{cases} \dfrac{2^{\alpha-1}+5}{3}, & \text{if } \alpha \text{ is odd,} \\ \dfrac{2^{\alpha-1}+4}{3}, & \text{if } \alpha \text{ is even.} \end{cases}$$

Proof. (1). It suffices to prove the following property: for any integers m, n with $\gcd(m,n) = 1$, we have

$$\mathrm{QR}(mn) = \mathrm{QR}(m)\mathrm{QR}(n).$$

Let $\mathcal{Q}_N := \{a \in \{0,1,\ldots,N-1\} \mid \exists\ x \ \text{ s.t. } \ x^2 \equiv a \pmod N\}$. We prove that the map

$$\begin{aligned} f : \mathcal{Q}_{mn} &\to \mathcal{Q}_m \times \mathcal{Q}_n \\ a &\to (a_1, a_2) \end{aligned}$$

is bijective, where a_1 is the unique element in $\mathcal{Q}_m$ satisfying $a \equiv a_1 \pmod{m}$, and a_2 is the unique element in $\mathcal{Q}_n$ satisfying $a \equiv a_2 \pmod{n}$. Since $x^2 \equiv a \pmod{mn}$ implies $x^2 \equiv a_1 \pmod{m}$ and $x^2 \equiv a_2 \pmod{n}$, we conclude that (a_1, a_2) is indeed an element of $\mathcal{Q}_m \times \mathcal{Q}_n$. Hence, f is well-defined as a map from $\mathcal{Q}_{mn}$ to $\mathcal{Q}_m \times \mathcal{Q}_n$. Injectivity of f follows directly from the Chinese Remainder Theorem (CRT), since a is uniquely determined by, and can be recovered from, the tuple (a_1, a_2). On the other hand, for any $(a_1, a_2) \in \mathcal{Q}_m \times \mathcal{Q}_n$, let $x_1^2 \equiv a_1 \pmod{m}$ and $x_2^2 \equiv a_2 \pmod{n}$. By the CRT, there exists an unique integer $x \in \{0, 1, \ldots, mn-1\}$ such that $x \equiv x_1 \pmod{m}$ and $x \equiv x_2 \pmod{n}$. Hence, $x^2 \equiv a_1 \pmod{m}$ and $x^2 \equiv a_2 \pmod{n}$. Again by the CRT, let $a \in \{0, 1, \ldots, mn-1\}$ be the unique integer satisfying $a \equiv a_1 \pmod{m}$ and $a \equiv a_2 \pmod{n}$. We have $x^2 \equiv a \pmod{mn}$, which implies $a \in \mathcal{Q}_{mn}$ and $f(a) = (a_1, a_2)$. Therefore, f is bijective, and hence $\mathrm{QR}(mn) = \mathrm{QR}(m)\mathrm{QR}(n)$.

(2). For $i \in \{0, 1, \ldots, \alpha-1\}$, consider the set $\mathcal{S}_i := \{a \in \{1, \ldots, p^\alpha - 1\} \mid p^i \parallel a\}$[22]. In particular, we define $\mathcal{S}_\alpha = \{0\}$. It is easy to see that $\{0, 1, \ldots, p^\alpha - 1\}$ is the disjoint union of $\mathcal{S}_0, \ldots, \mathcal{S}_\alpha$, and $|\mathcal{S}_i| = \varphi(p^{\alpha-i})$. Let $\mathcal{R}_i := \{x^2 \bmod p^\alpha \mid x \in \mathcal{S}_i\}$. It suffices to observe the following fact:
– Firstly, note that for every $\lceil \alpha/2 \rceil \leqslant i \leqslant \alpha$, we have $\mathcal{R}_i = \{0\}$.
– Secondly, for any i, j such that $\lceil \alpha/2 \rceil - 1 \geqslant i > j \geqslant 0$, we have $\mathcal{R}_i \cap \mathcal{R}_j = \varnothing$. Otherwise, assume $x = p^i x'$ and $y = p^j y'$, where both x' and y' are coprime to p, and assume $x^2 \equiv y^2 \pmod{p^\alpha}$. Then, we have $p^{2(i-j)} x'^2 \equiv y'^2 \pmod{p^{\alpha-2j}}$. Since $\alpha - 2j \geqslant 1$, it follows that $y'^2 \equiv 0 \pmod{p}$, contradicting $\gcd(y', p) = 1$.
– Finally, for every $0 \leqslant i \leqslant \lceil \alpha/2 \rceil - 1$, we have $|\mathcal{R}_i| = \frac{1}{2}\varphi(p^{\alpha-2i})$.
Assume $x = p^i x'$ and $y = p^i y'$ are two elements in $\mathcal{S}_i$, where both x' and y' lie in $\{1, \ldots, p^{\alpha-i} - 1\}$ and are coprime to p. Note that $x^2 = y^2 \pmod{p^\alpha} \iff x'^2 \equiv y'^2 \pmod{p^{\alpha-2i}} \iff x' - y' \equiv 0 \pmod{p^{\alpha-2i}}$ or $x' + y' \equiv 0 \pmod{p^{\alpha-2i}}$. This implies that, if the set $\{1, \ldots, p^{\alpha-i} - 1\}$ is partitioned into the following p^i subsets: $[0, p^{\alpha-2i} - 1]$, $[p^{\alpha-2i}, 2p^{\alpha-2i} - 1]$, $\ldots$, $[(p^i - 1)p^{\alpha-2i}, p^i \cdot p^{\alpha-2i} - 1]$, then each subset yields the same set of quadratic residues. Moreover, the elements in each subset can be partitioned into pairs, where each pair has the same quadratic residue. This proves that $|\mathcal{R}_i| = \frac{1}{2}\varphi(p^{\alpha-2i})$.
Combining the above three propositions, we immediately obtain

$$\mathrm{QR}(p^\alpha) = 1 + \frac{1}{2} \sum_{i=0}^{\lceil \alpha/2 \rceil - 1} \varphi(p^{\alpha-2i}).$$

(3). It suffices to prove that for any $\alpha \geqslant 3$, $\mathrm{QR}(2^\alpha) - \mathrm{QR}(2^{\alpha-2}) = 2^{\alpha-3}$. Let $\mathcal{S}_{\mathsf{odd}} := \{a \in \{0, 1, \ldots, 2^\alpha - 1\} \mid 2 \nmid a\}$, $\mathcal{S}_{\mathsf{even}} := \{a \in \{0, 1, \ldots, 2^\alpha - 1\} \mid 2 \mid a\}$ and let $\mathcal{R}_{\mathsf{odd}} := \{x^2 \bmod 2^\alpha \mid x \in \mathcal{S}_{\mathsf{odd}}\}, \mathcal{R}_{\mathsf{even}} := \{x^2 \bmod 2^\alpha \mid x \in \mathcal{S}_{\mathsf{even}}\}$. Note that for element $x = 2x' \in \mathcal{S}_{\mathsf{even}}$, its quadratic residue d must be divisible by 4. Assume $d = 4d'$, we have $x^2 \equiv d \pmod{2^\alpha} \iff x'^2 \equiv d' \pmod{2^{\alpha-2}}$, this implies $|\mathcal{R}_{\mathsf{even}}| = \mathrm{QR}(2^{\alpha-2})$. Next we proof that for any $\alpha \geqslant 3$, $\mathcal{R}_{\mathsf{odd}} = \{a \in \{0, 1, \ldots, 2^\alpha - 1\} \mid a \equiv 1 \pmod{8}\}$ (and hence $|\mathcal{R}_{\mathsf{odd}}| = 2^{\alpha-3}$). Every odd square is congruent to 1 modulo 8, so it remains to show that every integer congruent

[22] Here, $p^i \parallel a$ denotes that $p^i \mid a$ but $p^{i+1} \nmid a$.

to 1 modulo 8 is a quadratic residue modulo 2^α. We proceed by induction on α. The claim is clearly true for $\alpha = 3$. Assume that the claim holds for $\alpha = k \geqslant 3$. Then, for any a such that $a \equiv 1 \pmod 8$ and $a \in \{0, 1, \ldots, 2^k - 1\}$, there exists x such that $x^2 \equiv a \pmod{2^k}$. This implies $x^2 \equiv a$ or $a + 2^k \pmod{2^{k+1}}$. Note that $(x + 2^{k-1})^2 - x^2 \equiv 2^k x + 2^{2k-2} \pmod{2^{k+1}}$. Moreover, since x is odd and $2k - 2 \geqslant k + 1$, we have $(x + 2^{k-1})^2 - x^2 \equiv 2^k \pmod{2^{k+1}}$. This implies that either $x^2 \equiv a \pmod{2^{k+1}}$ and $(x + 2^{k-1})^2 \equiv a + 2^k \pmod{2^{k+1}}$, or $x^2 \equiv a + 2^k \pmod{2^{k+1}}$ and $(x + 2^{k-1})^2 \equiv a \pmod{2^{k+1}}$. Therefore, both a and $a + 2^k$ are quadratic residues modulo 2^{k+1}, and hence the claim holds for $\alpha = k+1$. Finally, since $\mathcal{R}_{\mathsf{even}} \cap \mathcal{R}_{\mathsf{odd}} = \emptyset$, we immediately obtain $\mathrm{QR}(2^\alpha) - \mathrm{QR}(2^{\alpha-2}) = 2^{\alpha-3}$.

Note that $\mathrm{QR}(2) = 2$ and $\mathrm{QR}(4) = 2$.

If $\alpha \geqslant 3$ is odd, it follows that

$$\begin{aligned}\mathrm{QR}(2^\alpha) &= (\mathrm{QR}(2^\alpha) - \mathrm{QR}(2^{\alpha-2})) + \cdots + (\mathrm{QR}(2^3) - \mathrm{QR}(2)) + \mathrm{QR}(2) \\ &= (2^{\alpha-3} + \cdots + 2^2 + 1) + 2 = \frac{2^{\alpha-1} + 5}{3}.\end{aligned}$$

If $\alpha \geqslant 3$ is even, it follows that

$$\begin{aligned}\mathrm{QR}(2^\alpha) &= (\mathrm{QR}(2^\alpha) - \mathrm{QR}(2^{\alpha-2})) + \cdots + (\mathrm{QR}(2^4) - \mathrm{QR}(2^2)) + \mathrm{QR}(2^2) \\ &= (2^{\alpha-3} + \cdots + 2^3 + 2) + 2 = \frac{2^{\alpha-1} + 4}{3}.\end{aligned}$$

Moreover, note that the formula also holds for $\alpha = 1$ and $\alpha = 2$. Therefore, Claim (3) holds. □