Constrained Verifiable Random Functions Without Obfuscation and Friends

Nicholas Brandt [1], Miguel Cueto Noval [2], Christoph U. Günther [2], Akin Ünal [2], and Stella Wohnig [3,4]

[1]ETH Zurich, crypto@nicholasbrandt.de
[2]Institute of Science and Technology Austria, {mcuetono, cguenthe, auenal}@ista.ac.at
[3]CISPA Helmholtz Institute for Information Security, stella.wohnig@cispa.de
[4]Saarland University

June 4, 2025

Abstract

CVRFs are PRFs that unify the properties of verifiable and constrained PRFs. Since they were introduced concurrently by Fuchsbauer and Chandran-Raghuraman-Vinayagamurthy in 2014, it has been an open problem to construct CVRFs without using heavy machinery such as multilinear maps, obfuscation or functional encryption.

We solve this problem by constructing a prefix-constrained verifiable PRF that does not rely on the aforementioned assumptions. Essentially, our construction is a verifiable version of the Goldreich-Goldwasser-Micali PRF. To achieve verifiability we leverage degree-2 algebraic PRGs and bilinear groups. In short, proofs consist of intermediate values of the Goldreich-Goldwasser-Micali PRF raised to the exponents of group elements. These outputs can be verified using pairings since the underlying PRG is of degree 2.

We prove the selective security of our construction under the *Decisional Square Diffie-Hellman* (DSDH) assumption and a new assumption, which we dub *recursive Decisional Diffie-Hellman* (recursive DDH).

We prove the soundness of recursive DDH in the generic group model assuming the hardness of the Multivariate Quadratic (MQ) problem and a new variant thereof, which we call MQ+.

Last, in terms of applications, we observe that our CVRF is also an exponent (C)VRF in the plain model. Exponent VRFs were recently introduced by Boneh et al. (Eurocrypt'25) with various applications to threshold cryptography in mind. In addition to that, we give further applications for prefix-CVRFs in the blockchain setting, namely, stake-pooling and compressible randomness beacons.

1 Introduction

Pseudorandom Functions (PRFs) allow for the generation of bit strings that are indistinguishable from true random coins, thereby constituting a central cryptographic building

block. Concretely, a PRF is a keyed *deterministic* function that has to be indistinguishable from a truly random function for any party that does not know the secret key of the PRF. Goldreich, Goldwasser & Micali [GGM86] showed how to construct PRFs from *Pseudorandom Generators* (PRGs). In contrast to PRFs, PRGs only need to generate a polynomial number of pseudorandom bits from a truly random key (or seed) and can be instantiated from one-way functions [Lev85, HILL99].

A popular line of research augments PRFs with additional properties. In this work, we will study two useful properties for PRFs: *constrainability* and *verifiability*.

A *Constrained PRF* (CPRF) [BW13, KPTZ13, BGI14] allows deriving constrained keys from the root secret key. While a constrained key enables one to evaluate the CPRF on a proper subset $\mathcal{X} \subsetneq \{0,1\}^\ell$ of its input domain, the security of the CPRF guarantees that a constrained key does not reveal anything about the CPRF on inputs outside $\mathcal{X}$. This allows one to delegate the power of the CPRF hierarchically.

By design, outputs of a PRF cannot be *trusted* by third parties, as checking the correctness of a potential output of a PRF requires access to its secret key. This limits the use-case of plain PRFs to inherently trusted settings. *Verifiable Random Functions* (VRFs) [MRV99] are PRFs that solve this problem by tying a public verification key to the secret key and accompanying outputs by proofs of correctness. Without a proof, any output of the VRF must be indistinguishable from random coins, even given the verification key. However, with the corresponding proof, correctness of the output can be easily checked. To be practical in trustless settings, VRFs must satisfy a very strong soundness property called *unique provability*, which asserts that even for malformed verification keys (and adversarially programmed random oracles) it is not possible to prove the correctness of two different output values (for the same input).

Since both primitives, CPRFs and VRFs, enjoy simple instantiations, one might assume that a natural construction could be made to achieve both *constrainability* and *verifiability*, at the same time. Quite surprisingly, however, the construction of PRFs that are both, constrained and verifiable, turned out to be quite involved. In 2014, Fuchsbauer [Fuc14] and Chandran, Raghuraman & Vinayagamurthy [CRV14] put forth the notion of *constrained verifiable PRFs* (CVRFs) that unify CPRFs and VRFs. CVRFs are versatile tools. For example, prefix-constrained CVRFs imply hierarchical identity-based signature algorithms with unique pseudorandom signatures. Both works devised candidates based on multilinear maps (mmaps), a heavy tool. Subsequent works [LLC15, LZW19, DDM17, ZLX23] tried to relax this assumption, but still relied on indistinguishability Obfuscation (iO) or functional encryption (FE) [Dat20].

Naturally, the heavy machinery of iO, mmaps and FE yield impractical constructions that are hard to understand and implement. Hence, we raise the following decade-long open question:

Can we construct constrained verifiable PRFs from simple cryptographic building blocks?

1.1 Our Candidate

In this work, we answer the above question positively. Concretely, we devise a candidate construction for prefix-constrained verifiable PRFs, which only makes use of bilinear groups and algebraic PRGs of degree 2. Those are PRGs whose outputs are computed by quadratic multivariate polynomials.

The Starting Point. Recall the (constrained) PRF construction of Goldreich, Goldwasser & Micali [GGM86]. It uses a length-doubling PRG $G\colon \{0,1\}^n \to \{0,1\}^{2n}$ where we let G_0 and G_1 be the functions computing the first and last n output bits of G, respectively. The value of the PRF F on a point $x = x_1 \cdots x_\ell \in \{0,1\}^\ell$ is given by

$$F_{\mathsf{sk}}(x) := G_x(\mathsf{sk}) := G_{x_\ell}(G_{x_{\ell-1}}(\cdots(G_{x_2}(G_{x_1}(\mathsf{sk})))))$$

where $\mathsf{sk} \leftarrow \{0,1\}^n$ is the secret key of the PRF. Given the simplicity of this construction, its security follows from a simple hybrid argument. In addition, F is already a prefix-constrained PRF [BW13, KPTZ13, BGI14]. Indeed, given a constrained key $\mathsf{sk}_w = G_w(\mathsf{sk})$ for a prefix $w \in \{0,1\}^{\ell'}$, one can compute any value $F_{sk}(wx')$, while the pseudorandomness of G implies that a value $F_{\mathsf{sk}}(x)$ still looks random as long as w is not a prefix of $x \in \{0,1\}^\ell$.

A First Attempt at Adding Verifiability. As a first step, we take the CPRF of Goldreich, Goldwasser & Micali [GGM86] and move it to an algebraic setting to allow for verifiability. To this end, let $e\colon \mathbb{G}_1^2 \to \mathbb{G}_2$ be a bilinear pairing of groups $\mathbb{G}_1 = \langle \mathsf{g}_1 \rangle$ and $\mathbb{G}_2 = \langle \mathsf{g}_2 \rangle$ of prime order p. Further, let $G\colon \mathbb{Z}_p^n \to \mathbb{Z}_p^{2n}$ now be a PRG that outputs values computed by quadratic multivariate polynomials over $\mathbb{Z}_p$. Again, we subdivide $G = (G_0, G_1)$ into two parts computing the left and right half of its output.[1] Since G is quadratic, it is possible to compute the *target* group elements $\mathsf{g}_2^{G_0(\boldsymbol{s})}, \mathsf{g}_2^{G_1(\boldsymbol{s})} \in \mathbb{G}_2^n$ when given *source* group elements $\mathsf{g}_1^{\boldsymbol{s}} = (\mathsf{g}_1^{s_1}, \ldots, \mathsf{g}_1^{s_n}) \in \mathbb{Z}_p^n$ by using the bilinear pairing e. In other words, we can evaluate the PRG G in the exponents of group elements, however, this will necessitate a transition from $\mathbb{G}_1$ to $\mathbb{G}_2$.

We draw the secret key sk of the CPRF F as a seed for G, i.e. $\mathsf{sk} \leftarrow\!\!\$\ \mathbb{Z}_p^n$. Consequently, F_{sk} outputs now vectors $F_{\mathsf{sk}}(x) = G_{x_{\ell-1}}(\cdots(G_{x_1}(\mathsf{sk}))) \in \mathbb{Z}_p^n$. To allow for verification, we introduce the verification key $\mathsf{vk} = \mathsf{g}_2^{\mathsf{sk}}$, a vector of group elements whose exponents are the elements of sk, and supply a proof π with every output $F_{\mathsf{sk}}(x)$. The proof π consists of a list of group elements

$$\pi_0 = \mathsf{g}_1^{\mathsf{sk}}, \pi_1 = \mathsf{g}_1^{G_{x_1}(\mathsf{sk})}, \pi_2 = \mathsf{g}_1^{G_{x_2}(G_{x_1}(\mathsf{sk}))}, \ldots, \pi_\ell = \mathsf{g}_1^{G_{x_\ell}(\cdots(G_{x_1}(\mathsf{sk})))}.$$

Given π, we can verify the correctness of the output $F_{\mathsf{sk}}(x)$ of our CPRF as follows: we first verify the soundness of the first vector of group elements $\pi_0 \in \mathbb{G}_1^n$ by checking $e(\pi_0, \mathsf{g}_1) = \mathsf{vk}$. Subsequently, we verify the correctness of π_i by asserting the equality $e(\pi_i, \mathsf{g}_1) = \mathsf{g}_2^{G_{x_i}(\cdots(G_{x_1}(\mathsf{sk})))}$, where we compute $\mathsf{g}_2^{G_{x_i}(\cdots(G_{x_1}(\mathsf{sk})))}$ by evaluating G_{x_i} at the exponents of π_{i-1}. Finally, we verify the correct evaluation of F_{sk} at x by ensuring that $\mathsf{g}_1^{F_{\mathsf{sk}}(x)} = \pi_\ell$ holds. This approach enables the uniqueness of output elements, even under a malformed verification key vk, proof π and PRG G. Thus, it fulfills the very strict requirement of *unique provability* which is necessary for VRFs. Additionally, the simplicity of this construction allows constraining a constrained key even further, fulfilling the notion of *delegability* [CRV14]. On top of that, constrained keys themselves can be verified and fulfil again unique provability.

[1] Because of technical reasons, we will actually choose a fresh PRG $G^{(i)} : \mathbb{Z}_p^n \to \mathbb{Z}_p^{2n}$ for every layer $i \in [\ell]$ of the binary tree underlying our construction later.

An Issue with the First Attempt and the Solution. While the first attempt is verifiable, a keen eye will have noticed that the above construction is *not* pseudorandom. To demonstrate this, consider the case $\ell = 2$. The proof $\pi = (\pi_0, \pi_1, \pi_2)$ for the output $F_{\mathsf{sk}}(00)$ contains the value $\pi_1 = \mathsf{g}_1^{G_0(\mathsf{sk})}$. It can be used to compute $\mathsf{g}_2^{G_1(G_0(\mathsf{sk}))}$, which suffices to verify the other output $F_{\mathsf{sk}}(01)$. In general, this leads to a simple attack where an adversary queries a proof for 0^ℓ and challenges the pseudorandomness of F at $0^{\ell-1}1$.

To solve this problem, we redefine the output of the CVRF F as

$$F_{\mathsf{sk}}(x) = \mathsf{g}_2^{\boldsymbol{y}^2} \in \mathbb{G}_2^n \quad \text{where} \quad \boldsymbol{y} = G_{x_\ell}(\cdots(G_{x_1}(\mathsf{sk}))).$$

($\boldsymbol{y}^2 = (y_1^2, \ldots, y_n^2)$ denotes the vector of squares of entries of $\boldsymbol{y} = (y_1, \ldots, y_n) \in \mathbb{Z}_p^n$.) This output can still be verified using π by checking that $F_{\mathsf{sk}}(x) = e(\pi_\ell, \pi_\ell)$ while intuitively ensuring pseudorandomness under the *Decisional Square Diffie-Hellman* (DSDH) assumption.[2] (An example for input lengths of $\ell = 2$ is given in Fig. 1.) Indeed, this

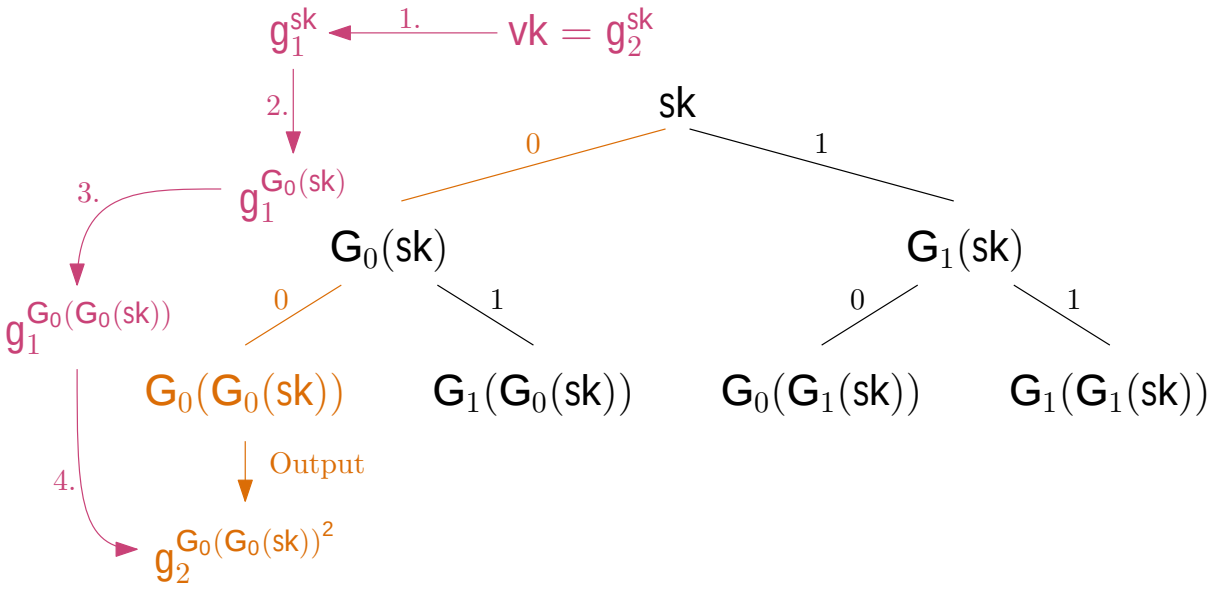

Figure 1: Illustration of our construction for 2 bit inputs. For the sake of example, it shows the evaluation on input 00 by highlighting the corresponding path and output in orange. The verification key vk, the proof $\pi = (\pi_0, \pi_1, \pi_2)$, and the four iterative verification steps are depicted in pink. All verification steps rely on the pairing e, especially steps 2 and 3, which evaluate the degree-2 PRG in the exponent using e.

change invalidates the given counterexample for $\ell = 2$, as DSDH implies that the output $F_{\mathsf{sk}}(01) = \mathsf{g}_2^{G_1(G_0(\mathsf{sk}))^2}$ stays pseudorandom even if an adversary can obtain $\mathsf{g}_2^{G_1(G_0(\mathsf{sk}))}$.

Unfortunately, fully proving the selective[3] security of the CVRF F remains challenging. The reason lies in the fact that F commits to the root secret key sk in two ways: by the verification key $\mathsf{vk} = \mathsf{g}_2^{\mathsf{sk}}$, a vector of group elements, and by constrained keys

[2]The decisional square Diffie-Hellman problem (in the target group $\mathbb{G}_2$) consists in distinguishing $(\mathsf{g}_2, \mathsf{g}_2^a, \mathsf{g}_2^{a^2})$ and $(\mathsf{g}_2, \mathsf{g}_2^a, \mathsf{g}_2^r)$ for $a, r \leftarrow\!\!\$\ \mathbb{Z}_p$.

[3]In this work, we focus only on the selective security of CVRFs, where an adversary has to commit to a challenge point $x^* \in \{0,1\}^\ell$ before it sees a verification key. Proving the adaptive security is out of scope as it requires intricate rewinding techniques [HKK23].

$G_w(\mathsf{sk})$, vectors over $\mathbb{Z}_p$. This prohibits typical reduction techniques of, for example, the generic group model, and makes it hard to hop between different games by using the pseudorandomness of G. Indeed, sk is committed to in such a rigid way that it is impossible for a reduction to directly replace some values of F by randomness behind the scenes without the adversary noticing.

1.2 Contribution

Proving the pseudorandomness of the CVRF F is non-trivial and requires new techniques. As a first step, we modularize the proof by a new tool, which we dub the *recursive Decisional Diffie-Hellman* (recursive DDH) assumption. This assumption (detailed in the technical overview) asserts that an *a posteriori* challenge must be hard if an *a priori* challenge is hard. From this assumption and DSDH, the pseudorandomness of F follows, leading to our first main result:

Theorem 1. *Under the decisional square Diffie-Hellman and the recursive decisional Diffie-Hellman assumption, F is a selectively secure prefix-constrained verifiable random function.*

As a next step and second result, we prove the soundness of recursive DDH in Maurer's *Generic Group Model* [Mau05] assuming the hardness of MQ and MQ+. MQ is a standard assumption capturing the hardness of solving multivariate quadratic equation systems. For example, multiple NIST post-quantum digital signatures candidates are based on it. MQ+ is a new assumption that we introduce and that is closely related to MQ. We will discuss MQ and MQ+ in the technical overview, and analyze attacks on recursive DDH, MQ and MQ+ in App. A.

Theorem 2. *In the generic group model, the recursive DDH assumption is implied by the MQ and the MQ+ assumption.*

Finally, we adapt the verification algorithm to improve the verification time of our CVRF construction. For $n = \ell = 256$, we estimate that verification takes ca. 1 minute[4], which is reasonably practical. Our optimization reduces the number of pairings computed by the verifier to $O(\ell n)$. It does so by randomly combining n quadratic equations into one. This introduces a negligible error probability that only depends on the verifier's coins. Thus, it does not go against the spirit of unique provability. For details see App. B.

1.3 Technical Overview

We will first sketch the security of our CVRF F under our novel recursive DDH assumption, and then prove the hardness of recursive DDH in the generic group model under two falsifiable and contained assumptions, MQ and MQ+.

Fresh PRGs per Layer.. Before we sketch our proofs of the pseudorandomness of F, we point out an additional technicality. In our actual construction of F, we use a fresh algebraic degree-2 PRG $G^{(i)} = (G_0^{(i)}, G_1^{(i)}) : \mathbb{Z}_p^n \to \mathbb{Z}_p^{2n}$ for every layer $i \in [\ell]$. This means,

[4]We assume that each pairing takes 1 millisecond and neglect all other operations at verification.

the coefficients of the quadratic polynomials computing the functions $G^{(1)}, \ldots, G^{(\ell)}$ are all distributed uniformly and independently at random. Consequently, we define the value of F at input $x = x_1 \cdots x_\ell$ by

$$F_{\mathsf{sk}}(x) := \mathsf{g}_2^{\boldsymbol{y}^2} \text{ where } \boldsymbol{y} = G_x(\mathsf{sk}) := G^{(\ell)}_{x_\ell}(G^{(\ell-1)}_{x_{\ell-1}}(\cdots(G^{(2)}_{x_2}(G^{(1)}_{x_1}(\mathsf{sk}))))),$$

with $\boldsymbol{y}^2 = (y_1^2, \ldots, y_n^2)$. Constrained keys for prefixes $w = w_1 \cdots w_{\ell'}$ are computed by

$$\mathsf{sk}_w := G_w(\mathsf{sk}) = G^{(\ell')}_{w_{\ell'}}(G^{(\ell'-1)}_{w_{\ell'-1}}(\cdots(G^{(2)}_{w_2}(G^{(1)}_{w_1}(\mathsf{sk}))))).$$

Because of this change, we can always assume in the selective security game that an adversary submits 0^ℓ as challenge point x^*, at which it has to distinguish the value of F from uniform random group elements. Indeed, if $x^* \neq 0^\ell$, we can simply swap $G_0^{(i)}$ and $G_1^{(i)}$ whenever $x_i^* \neq 0$.

The Recursive Decisional Diffie-Hellman Assumption. The *recursive Decisional Diffie-Hellman* (recursive DDH) assumption states that, for each sampler Samp, a certain challenge stays hard even if we hand out more, seemingly useless, auxiliary information. Concretely, let $e\colon \mathbb{G}_1^2 \to \mathbb{G}_2$ be a bilinear map of groups and Samp be a PPT algorithm that on input $1^\lambda, p, n$ outputs three *hint* functions $h_0\colon \mathbb{Z}_p^n \to \mathbb{Z}_p^{m_0}$, $h_1\colon \mathbb{Z}_p^n \to \mathbb{Z}_p^{m_1}$, $h_2\colon \mathbb{Z}_p^n \to \mathbb{Z}_p^{m_2}$ and a *challenge* function $c\colon \mathbb{Z}_p^n \to \mathbb{Z}_p^n$.

The recursive DDH assumption states for each sampler Samp that, if $\mathsf{g}_2^{c(\boldsymbol{s})}$ is pseudorandom given

$$(c, h_0, h_1, h_2), \qquad h_0(\boldsymbol{s}), \qquad \mathsf{g}_1^{h_1(\boldsymbol{s})}, \qquad \mathsf{g}_2^{h_2(\boldsymbol{s})}, \qquad \mathsf{g}_2^{\boldsymbol{s}},$$

then $\mathsf{g}_2^{c(\boldsymbol{s}')}$ is pseudorandom given

$$\begin{array}{lrrrrr} & (c, h_0, h_1, h_2), & h_0(\boldsymbol{s}'), & \mathsf{g}_1^{h_1(\boldsymbol{s}')}, & \mathsf{g}_2^{h_2(\boldsymbol{s}')}, & \mathsf{g}_2^{\boldsymbol{s}'}, \\ \textbf{and} & (G_0, G_1), & G_1(\boldsymbol{x}), & \mathsf{g}_1^{\boldsymbol{x}}, & & \end{array}$$

where we draw $\boldsymbol{s}, \boldsymbol{x} \leftarrow \mathbb{Z}_p^n$, $(c, h_0, h_1, h_2) \leftarrow \mathsf{Samp}(1^\lambda, p, n)$ and set $\boldsymbol{s}' = G_0(\boldsymbol{x})$ for two uniformly random maps $G_0, G_1 : \mathbb{Z}_p^n \to \mathbb{Z}_p^n$ computed by multivariate quadratic polynomials. We will call the first decisional problem the if-*challenge*, and the second decisional problem the then-*challenge*.

Let us explain the soundness of this assumption intuitively: The if-challenge states that the auxiliary information $h_0(\boldsymbol{s}), \mathsf{g}_1^{h_1(\boldsymbol{s})}, \mathsf{g}_2^{h_2(\boldsymbol{s})}$ and $\mathsf{g}_2^{\boldsymbol{s}}$ do not help at distinguishing $\mathsf{g}_2^{c(\boldsymbol{s})}$ from a vector of n random group elements of $\mathbb{G}_2$. This implies, in particular, that $\mathsf{g}_2^{c(\boldsymbol{s})}$, while it is correlated with the auxiliary information, cannot be simply algebraically derived from the other group elements $\mathsf{g}_1^{h_1(\boldsymbol{s})}, \mathsf{g}_2^{h_2(\boldsymbol{s})}, \mathsf{g}_2^{\boldsymbol{s}}$, even when knowing $h_0(\boldsymbol{s})$ in plain. The then-part of the assumption postulates that $\mathsf{g}_2^{c(\boldsymbol{s})}$ stays pseudorandom, even if $\boldsymbol{s} = G_0(\boldsymbol{x})$ is now sampled as the left-hand output of the PRG G, and even if we additionally give the adversary the corresponding right-hand output $G_1(\boldsymbol{x})$ in plain and the seed of the PRG $\mathsf{g}_1^{\boldsymbol{x}}$ as source group elements. Hence, recursive DDH simply states that a certain group vector stays pseudorandom even if we hand out more information, $\mathsf{g}_1^{\boldsymbol{x}}$ and $G_1(\boldsymbol{x})$,

to the distinguisher. Note if we would hand out only one of $\{\mathsf{g}_1^{\boldsymbol{x}}, G_1(\boldsymbol{x})\}$, the correctness of the assumption would easily follow.[5]

Unfortunately, giving a reduction of the if-problem to the then-problem becomes challenging as soon as one hands out both additional auxiliary elements $\mathsf{g}_1^{\boldsymbol{x}}$ and $G_1(\boldsymbol{x})$. While it is intuitively clear that both vectors together do not help at distinguishing $\mathsf{g}_2^{c(\boldsymbol{s})}$ from random group elements, the above techniques fail at proving this, since we are committed to the PRG-seed $\boldsymbol{x}$ in two different ways (as group elements $\mathsf{g}_1^{\boldsymbol{x}}$ and as plain numbers $G_1(\boldsymbol{x})$). For now, we postpone the discussion of how we solve this conundrum and continue to show the application of recursive DDH.

Proving Security under Recursive DDH. The recursive DDH assumption gives an elegant way to prove the pseudorandomness of our CVRF F. Let $\mathcal{A}$ be a selective adversary that, at the start of the game, sends a challenge point $x^* \in \{0,1\}^\ell$ to a challenger $\mathcal{C}$. As explained above, we can assume $x^* = 0^\ell$ without loss of generality. $\mathcal{C}$ has to respond by sending $\mathcal{A}$ a verification key vk, constrained[6] keys $\mathsf{sk}_1, \mathsf{sk}_{01}, \ldots, \mathsf{sk}_{0^{\ell-2}1}$ (for all non-prefixes of 0^ℓ), the output value $F_{\mathsf{sk}}(0^{\ell-1}1)$, and finally proofs $\pi_1, \ldots, \pi_\ell$ for the correctness of the aforementioned constrained keys and output value. Here, $\mathsf{sk} = \boldsymbol{s} \leftarrow \mathbb{Z}_p^n$ is a uniformly random vector. The verification key is given by $\mathsf{vk} = \mathsf{g}_2^{\boldsymbol{s}}$. The output at $0^{\ell-1}1$ is given by

$$F_{\mathsf{sk}}(0^{\ell-1}1) = \mathsf{g}_2^{\boldsymbol{y}^2} \quad \text{for} \quad \boldsymbol{y} = G_1^{(\ell)}(G_0^{(\ell-1)}(\cdots(G_0^{(1)}(\boldsymbol{s})))).$$

The constrained keys are all vectors in $\mathbb{Z}_p^n$ of shape

$$G_1^{(1)}(\boldsymbol{s}), \quad G_1^{(2)}(G_0^{(1)}(\boldsymbol{s})), \quad G_1^{(3)}(G_0^{(2)}(G_0^{(1)}(\boldsymbol{s}))), \quad \ldots, \quad G_1^{(\ell-1)}(G_0^{(\ell-2)}(\cdots(G_0^{(1)}(\boldsymbol{s})))),$$

while the proof elements are vectors of group elements given by

$$\pi_0 = \mathsf{g}_1^{\boldsymbol{s}}, \quad \pi_1 = \mathsf{g}_1^{G_0^{(1)}(\boldsymbol{s})}, \quad \ldots, \quad \pi_{\ell-1} = \mathsf{g}_1^{G_0^{(\ell-1)}(\cdots(G_0^{(1)}(\boldsymbol{s})))}, \quad \pi_\ell = \mathsf{g}_1^{G_1^{(\ell)}(G_0^{(\ell-1)}(\cdots(G_0^{(1)}(\boldsymbol{s}))))}.$$

Let us first consider the base-case of $\ell = 0$. In this case, our CVRF has exactly one possible input (the empty word ϵ), and $\mathcal{A}$ only gets the verification key $\mathsf{g}_2^{\boldsymbol{s}}$, while it has to distinguish the only possible output value $F_{\mathsf{sk}}(\epsilon) = \mathsf{g}_2^{\boldsymbol{s}^2}$ from random group elements.[7] Obviously, the pseudorandomness of our CVRF for $\ell = 0$ is equivalent to the decisional square Diffie-Hellman problem and, therefore, follows directly from the corresponding assumption.

The pseudorandomness of our construction for $\ell = 1$ follows now by applying recursive DDH. Indeed, by letting $c(\boldsymbol{s}) = \boldsymbol{s}^2$ be the squaring function and defining all hint functions h_0, h_1, h_2 to be empty (and letting $G^{(\ell)} = (G_0^{(\ell)}, G_1^{(\ell)})$ be the PRG randomly sampled by the

[5] For example, if the adversary only receives $G_1(\boldsymbol{x})$ as additional auxiliary information, we can invoke (in the absence of $\mathsf{g}_1^{\boldsymbol{x}}$) the pseudorandomness of $G = (G_0, G_1)$, and replace $\boldsymbol{s} = G_0(\boldsymbol{x})$ and $G_1(\boldsymbol{x})$ in the distribution by uniformly random vectors $\boldsymbol{r}_1, \boldsymbol{r}_2 \leftarrow\!\!\$\ \mathbb{Z}_p^n$, which immediately reduces the if-challenge to the then-challenge.

[6] Note that from those values, $\mathcal{A}$ can compute all other possible queries on its own. Further, since constrained keys, evaluation values and proofs are generated deterministically, $\mathcal{A}$ does not profit from querying the same value multiple times.

[7] The proof of the correctness of $F_{\mathsf{sk}}(\epsilon)$ is given by the group element vector $\mathsf{g}_1^{\boldsymbol{s}}$. However, since $\ell = 0$, $\mathcal{A}$ can challenge the pseudorandomness of the CVRF at one point and, hence, not receive this proof element.

assumption), recursive DDH implies directly that $F_{\mathsf{sk}}(0) = \mathsf{g}_2^{G_0^{(\ell)}(\boldsymbol{x})^2}$ is pseudorandom given $G_1^{(\ell)}(\boldsymbol{x})$ and $\pi_{\ell-1} = \mathsf{g}_1^{\boldsymbol{x}}$. Since $\mathcal{A}$ can compute $\mathsf{vk} = \mathsf{g}_2^{\boldsymbol{x}}$, $\pi_\ell = \mathsf{g}_1^{G_1^{(\ell)}(\boldsymbol{x})}$ and $F_{\mathsf{sk}}(1) = \mathsf{g}_2^{G_1^{(\ell)}(\boldsymbol{x})^2}$ from this data, recursive DDH implies now the pseudorandomness of our construction for input lengths $\ell = 1$. (Note that the secret key is now called $\boldsymbol{x}$ instead of $\boldsymbol{s}$.)

Applying recursive DDH a second time, now with $\ell = 2$, $h_0(\boldsymbol{s})$ being the empty function, $h_1(\boldsymbol{s}) = G_1^{(\ell)}(\boldsymbol{s})$, $h_2(\boldsymbol{s}) = G_1^{(\ell)}(\boldsymbol{s})^2$ and $c(\boldsymbol{s}) = G_0^{(\ell)}(\boldsymbol{s})^2$, yields that $F_{\mathsf{sk}}(00) = \mathsf{g}_2^{c(\boldsymbol{s})} = \mathsf{g}_2^{c(G_0^{(\ell-1)}(\boldsymbol{x}))} = \mathsf{g}_2^{G_0^{(\ell)}(G_0^{(\ell-1)}(\boldsymbol{x}))^2}$ stays pseudorandom, for $\boldsymbol{s} = G_0^{(\ell-1)}(\boldsymbol{x})$, even when given

$$\pi_\ell = \mathsf{g}_1^{h_1(\boldsymbol{s})} = \mathsf{g}_1^{G_1^{(\ell)}(G_0^{(\ell-1)}(\boldsymbol{x}))}, \qquad F_{\mathsf{sk}}(01) = \mathsf{g}_2^{h_2(\boldsymbol{s})} = \mathsf{g}_2^{G_1^{(\ell)}(G_0^{(\ell-1)}(\boldsymbol{x}))^2},$$
$$\mathsf{sk}_1 = G_1^{(\ell-1)}(\boldsymbol{x}), \qquad \pi_{\ell-1} = \mathsf{g}_1^{\boldsymbol{x}}.$$

Again, $\mathcal{A}$ can compute the remaining element $\mathsf{vk} = \mathsf{g}_2^{\boldsymbol{x}}$ on its own, hence, the pseudorandomness of our CVRF for $\ell = 2$. Applying recursive DDH a third time will imply the pseudorandomness of our CVRF for inputs of length $\ell = 3$, and so on. Hence, the security of our CVRF candidate in the selective pseudorandomness game for inputs of length ℓ follows directly by the decisional square Diffie-Hellman assumption and ℓ applications of our recursive DDH assumption.

Proving Recursive DDH in the Generic Group Model. We will sketch now how to prove the soundness of the recursive DDH assumption in Maurer's Generic Group Model (GGM) [Mau05].[8] First, let us give a high-level overview: We perform two game hops to remove the additional auxiliary information given in the then-challenge. The first game hop replaces $\mathsf{g}_1^{\boldsymbol{x}}$ by dummy-elements. Since we are committed to $\mathsf{g}_1^{\boldsymbol{x}}$ by $G_1(\boldsymbol{x})$ and $\mathsf{g}_2^{\boldsymbol{s}} = \mathsf{g}_2^{G_0(\boldsymbol{x})}$, we cannot use the standard GGM technique of replacing $\mathsf{g}_1^{\boldsymbol{x}}$ by uniformly random group elements. Instead, we answer group queries using a carefully designed algebraic procedure. Omitting details for now, it allows us to invoke MQ+ to argue that the adversary cannot detect this change. After removing $\mathsf{g}_1^{\boldsymbol{x}}$, we perform a second game hop and replace $\boldsymbol{s} = G_0(\boldsymbol{x})$ and $G_1(\boldsymbol{x})$ by uniform randomness. The indistinguishability of both games follows directly from the MQ assumption. In the last game, $\boldsymbol{s}$ has been sampled uniformly at random and $\mathsf{g}_1^{\boldsymbol{x}}, G_1(\boldsymbol{x})$ have been removed. Now, we finish the security proof of recursive DDH by reducing the if-problem to this game.

Next, we introduce and discuss the MQ and MQ+ assumptions, and then give a more elaborated proof sketch of the soundness of recursive DDH in the GGM under MQ and MQ+.

MQ and MQ+. The decisional *Multivariate Quadratic* (MQ) problem is a post-quantum assumption on the hardness of solving random systems of multivariate quadratic equations. It is used to construct non-interactive zero-knowledge proofs [DJJ24] and underlies the security of a large proportion of candidates [BCD+24, AFI+24, BCC+24, BBFR24]

[8] Remember that in the GGM a generic adversary obtains handles for group elements, which only allow it to perform group and pairing operations. In Maurer's GGM, those handles are deterministic and given by increasing register addresses. As alternative model, one can also consider Shoup's GGM [Sho97] where handles are uniformly random bit strings.

in NIST's standardization process for additional post-quantum signatures. MQ+ is a computational assumption, similar to MQ, that requires an adversary to extract a simple non-trivial relationship about the solution of a random quadratic equation system. While the MQ+ problem is likely not equivalent to the MQ problem, we will present some intuitive arguments in App. A.3 why MQ+ should be hard whenever MQ is hard. Essentially, if there would be an MQ+ solver that outputs fresh solutions for MQ+, we could leverage it to a solving algorithm for MQ (under some assumptions).

For our construction, we need to assume, at a minimum, that G is a PRG. Indeed, otherwise an adversary could invert $G_1(\boldsymbol{x})$ and break the then-challenge, while the if-challenge might still be intractable for it. Note that we sample G as a function where each output is computed by a uniformly random quadratic polynomial $g \in \mathbb{Z}_p[X_1, \ldots, X_n]$ over $\mathbb{Z}_p$ in n variables. Distinguishing outputs of G from uniform randomness constitutes exactly the decisional MQ problem.

Now, the *MQ+* problem is intuitively similar. Formally, the MQ+ assumption states that it is hard for a PPT adversary, when given $2n$ uniformly random degree-2 polynomials $g_1, \ldots, g_{2n} \in \mathbb{Z}_p[X_1, \ldots, X_n]$, which compute $G : \mathbb{Z}_p^n \to \mathbb{Z}_p^{2n}$ for example, and their values $g_1(\boldsymbol{x}), \ldots, g_{2n}(\boldsymbol{x})$ on a secret point $\boldsymbol{x} \leftarrow\!\!\$\ \mathbb{Z}_p^n$, to find an additional quadratic polynomial $h \in \mathbb{Z}_p[X_1, \ldots, X_n]$ that vanishes on $\boldsymbol{x}$ but is linearly independent of $g_1(X) - g_1(\boldsymbol{x}), \ldots, g_{2n}(X) - g_{2n}(\boldsymbol{x})$. Let us explain this: when given $g_1, \ldots, g_{2n}$ and $g_1(\boldsymbol{x}), \ldots, g_{2n}(\boldsymbol{x})$, it is trivial to find new polynomials that vanish on $\boldsymbol{x}$, even if we do not know $\boldsymbol{x}$. Indeed, any linear combination $h(X) = \alpha_1 \cdot (g_1(X) - g_1(\boldsymbol{x})) + \cdots + \alpha_{2n} \cdot (g_{2n}(X) - g_{2n}(\boldsymbol{x}))$ must be zero at $\boldsymbol{x}$. On the other hand, finding a degree-2 polynomial h that vanishes on $\boldsymbol{x}$ and does not lie in the space spanned by $g_1(X) - g_1(\boldsymbol{x}), \ldots, g_{2n}(X) - g_{2n}(\boldsymbol{x})$ would give us a new and non-trivial information about the solution of the equation system $g_1(X) = g_1(\boldsymbol{x}), \ldots, g_{2n}(X) = g_{2n}(\boldsymbol{x})$. Indeed, if we would be able to solve the MQ+ problem $O(n^2)$ times with fresh and independent solutions $h_1, h_2 \ldots$, we could simply linearize the quadratic equation system and solve it trivially. We will postpone cryptanalysis of MQ and MQ+ for now, and continue with showing how both assumptions imply recursive DDH in the generic group model.

Proof Sketch. Let Samp be an efficient probabilistic sampling algorithm that on input $1^\lambda, p, n$ outputs functions $c\colon \mathbb{Z}_p^n \to \mathbb{Z}_p^n$, $h_0\colon \mathbb{Z}_p^n \to \mathbb{Z}_p^{m_0}$, $h_1\colon \mathbb{Z}_p^n \to \mathbb{Z}_p^{m_1}$, $h_2\colon \mathbb{Z}_p^n \to \mathbb{Z}_p^{m_2}$. To prove the soundness of the assumption, we need to reduce the hardness of the if-challenge, which consists of distinguishing $\mathsf{g}_2^{c(\boldsymbol{s})}$ from random group elements when given $h_0(\boldsymbol{s})$, $\mathsf{g}_1^{h_1(\boldsymbol{s})}$ and $\mathsf{g}_2^{h_2(\boldsymbol{s})}$, to the hardness of the then-challenge, which consists of distinguishing $\mathsf{g}_2^{c(\boldsymbol{s})}$ from random group elements when given $h_0(\boldsymbol{s})$, $\mathsf{g}_1^{h_1(\boldsymbol{s})}$, $\mathsf{g}_2^{h_2(\boldsymbol{s})}$ and additionally $\mathsf{g}_1^{\boldsymbol{x}}$, $G_1(\boldsymbol{x})$ where $\boldsymbol{s} = G_0(\boldsymbol{x})$ (for a uniformly random quadratic map $G = (G_0, G_1)\colon \mathbb{Z}_p^n \to \mathbb{Z}_p^{2n}$).

Let G_0 be the game between an adversary $\mathcal{A}$ and a then-challenger Ch. To prove the recursive DDH assumption, we need to change G_0 to a game to which the if-challenge can be directly reduced.

Step 1: Removing $\mathsf{g}_1^{\boldsymbol{x}}$. As a first step, we would like to replace the group elements $\mathsf{g}_1^{\boldsymbol{x}}$ given to the adversary by random group elements or dummy elements, which are devoid of any information about $\boldsymbol{x}$. The problem, however, is that we are committed to $\boldsymbol{x}$ in two different ways: by $\mathsf{g}_2^{\boldsymbol{s}} = \mathsf{g}_2^{G_0(\boldsymbol{x})}$ and by $G_1(\boldsymbol{x})$. Indeed, replacing $\mathsf{g}_1^{\boldsymbol{x}}$ naively by group elements $\mathsf{g}_1^{\boldsymbol{x}'}$, for $\boldsymbol{x} \neq \boldsymbol{x}' \in \mathbb{Z}_p^n$, can directly be detected by $\mathcal{A}$ by applying G_0 or G_1 in the

exponent of $\mathsf{g}_1^{\boldsymbol{x}'}$ and comparing the result with $\mathsf{g}_2^{G_0(\boldsymbol{x})}$ or $G_1(\boldsymbol{x})$.

To solve this problem, let $g_1, \ldots, g_{2n} \in \mathbb{Z}_p[X_1, \ldots, X_n]$ be the quadratic polynomials computing G. Additionally, since $\mathcal{A}$ is generic, we model each of its group queries by polynomials on the exponents of its given group elements. This means, $\mathcal{A}$ can only submit group queries in the form of polynomials f to its group oracle. Whenever $\mathcal{A}$ submits such a query f, Ch has to tell $\mathcal{A}$ if f does vanish on the exponents of the given group elements. Given the bilinearity of the involved groups, all of those polynomials are quadratic.

Our novel technique for the first hybrid step consists in replacing $\mathsf{g}_1^{\boldsymbol{x}} = (\mathsf{g}_1^{x_1}, \ldots, \mathsf{g}_1^{x_n})$ by dummy-elements $\mathsf{g}_1^X = (\mathsf{g}_1^{X_1}, \ldots, \mathsf{g}_1^{X_n})$ and by devising an *algebraic algorithm* for answering group queries of the adversary. Concretely, in this new game, which we call G_1, Ch tells $\mathcal{A}$ that an equality-check[9] $f \in \mathbb{Z}_p[X_1, \ldots, X_n]$ passes on the exponents of $\mathsf{g}_1^{\boldsymbol{x}}$, i.e. $f(\boldsymbol{x}) = 0$, if it lies in the $\mathbb{Z}_p$-vector space

$$V := \mathrm{span}_{\mathbb{Z}_p}[g_1(X) - s_1, \ldots, g_n(X) - s_n, g_{n+1}(X) - w_1, \ldots, g_{2n}(X) - w_n]$$

spanned by the polynomials $g_1(X) - s_1, \ldots, g_n(X) - s_n$, $g_{n+1}(X) - w_1, \ldots, g_{2n}(X) - w_n \in \mathbb{Z}_p[X]$ where $\boldsymbol{s} = G_0(\boldsymbol{x})$ denotes the left-hand and $\boldsymbol{w} := G_1(\boldsymbol{x})$ the right-hand output of $G(\boldsymbol{x})$.

To prove that $\mathcal{A}$ cannot distinguish between G_0 and G_1, we invoke the MQ+ assumption. Indeed, $\mathcal{A}$ can only distinguish between both games if it can come up with a query f that passes in one game and fails in the other one. Since $(\boldsymbol{s}, \boldsymbol{w}) = G(\boldsymbol{x})$, we must have $f(\boldsymbol{x}) = 0$ whenever $f \in V$. Hence, to notice a difference, $\mathcal{A}$ must output a query f that passes in G_0 ($f(\boldsymbol{x}) = 0$), but fails in G_1 ($f \notin V$). Such a polynomial f is exactly a solution to the MQ+ problem with respect to $G = (G_0, G_1)$, which asks for a non-trivial degree-2 polynomial that vanishes on $\boldsymbol{x}$. Assuming the hardness of MQ+, we can deduce that $\mathcal{A}$ can not distinguish between the *honest* way of answering group queries in G_0 and our novel algebraic way of answering queries in G_1.

Step 2: Removing $G_1(\boldsymbol{x})$. In G_2, we switch $(\boldsymbol{s}, \boldsymbol{w})$ from $(G_0(\boldsymbol{x}), G_1(\boldsymbol{x}))$ to two uniformly random vectors in $\mathbb{Z}_p^n$. Indistinguishability follows directly from the normal MQ assumption. Indeed, after Step 1, all group elements given to $\mathcal{A}$ (and thus all information $\mathcal{A}$ can determine by generic group queries) do not depend on $\boldsymbol{x}$, but only on $(G_0(\boldsymbol{x}), G_1(\boldsymbol{x}))$.

Step 3: Reducing the if-Challenge to G_2. Finally, in G_2, $\boldsymbol{w}$ is uniformly random and independent of $\boldsymbol{s}$ and all other information given to $\mathcal{A}$. Additionally, $\mathcal{A}$ is given simulated group elements g_1^X instead of the real group elements $\mathsf{g}_1^{\boldsymbol{x}}$. At this point, we can reduce the if-challenge to G_2 by simulating all extra information ($\mathsf{g}_1^{\boldsymbol{x}}$ and $G_1(\boldsymbol{x})$) the then-adversary $\mathcal{A}$ would receive. There are some technicalities involved in how group queries of $\mathcal{A}$ are handled, as a reduction does not know $\boldsymbol{s}$ in plain. However, these issues can be solved by algebraically decomposing the group queries of $\mathcal{A}$. For all technical details, we refer to the full proof given in § 5.

[9] Note that $\mathcal{A}$'s equality checks must actually involve more variables. In fact, besides $X_1, \ldots, X_n$, f should contain variables $S_1, \ldots, S_n$ representing the exponents of $\mathsf{g}_2^{\boldsymbol{s}} \in \mathbb{G}_2^n$, variables $C_1, \ldots, C_n$ representing the exponents of $\mathsf{g}_2^{c(\boldsymbol{s})} \in \mathbb{G}_2^n$, variables $H_1^{(1)}, \ldots, H_{m_1}^{(1)}$ representing the exponents of $\mathsf{g}_1^{h_1(\boldsymbol{s})} \in \mathbb{G}_1^{m_1}$ and variables $H_1^{(2)}, \ldots, H_{m_2}^{(2)}$ representing the exponents of $\mathsf{g}_2^{h_2(\boldsymbol{s})} \in \mathbb{G}_2^{m_2}$. To keep this exposition simple, we neglect this fact and consider here group queries of $\mathcal{A}$ that only involve the elements $\mathsf{g}_1^{\boldsymbol{x}}$.

Cryptanalysis of Our Assumptions. Security hinges on our new recursive DDH assumption. In this work, recursive DDH only functions as an intermediate step to prove the security of our construction, and we can prove the soundness of our assumption in an idealized model under MQ and MQ+.

While MQ is a well-studied standard assumption in cryptography, MQ+ is a novel variation of it. While there is no direct reduction from MQ to MQ+, the hardness of the MQ+ problem should be comparable to the hardness of the MQ problem. Indeed, known algebraic attacks should not be faster on MQ+ than on MQ, and any efficient algorithm for solving MQ+ would, intuitively, imply a fast solving algorithm for MQ. We refer to App. A for detailed cryptanalytic discussions on recursive DDH, MQ and MQ+.

Modularizing the Underlying PRG Family. Note that the CVRF F uses uniformly random quadratic maps over $\mathbb{Z}_p$ as PRGs. It is possible to replace this distribution of PRGs by other, more specialized distributions of quadratic maps $\mathbb{Z}_p^n \to \mathbb{Z}_p^{2n}$. This may be more beneficial for practical considerations.

In App. D, we explain which properties a general distribution of degree-2 maps needs to have for our security proofs to work through. Specifically, we replace the MQ and MQ+ assumptions with properties of the underlying PRG distribution—namely, pseudorandomness and a new property we call *resistance*.

On Generic Constructions. Finally, let us address the question if it is possible to construct CVRFs generically from Non-Interactive Zero-Knowledge proofs (NIZKs) and CPRFs. First, we want to point out that NIZKs and other primitives depending on random oracles or common reference strings, are infeasible for this task, as the very strict requirement of unique provability needs to hold, even for *malformed* oracles and reference strings. However, in the plain model, one may still ask if it is possible to enhance CPRFs with verifiability by using non-interactive witness-indistinguishable proofs (NIWIs). Prior works [Bit20, GHKW17] constructed VRFs from NIWIs, puncturable PRFs and commitments. It is possible to extend these constructions to CVRFs, however, with significant drawbacks on security (and efficiency given the sizes of the involved NIWI proofs). The crux lies in the notion of *constraint-hiding*, introduced by Fuchsbauer [Fuc14]. Constraint-hiding is a technical property that allows us to significantly simplify the security game for CVRFs. A NIWI-based CVRF (following the blueprint of [Bit20, GHKW17]) not only fails to be constraint-hiding, it is also not possible to prove it secure in more elaborate security games that better reflect reality. We discuss this in full detail in App. C.

1.4 Applications

Exponent VRFs. Recently, Boneh et al. [BHLS25] introduced exponent VRFs (eVRFs). While a standard VRF $F_{\mathsf{sk}}(x) = y$ proves correctness of y, an eVRF proves correctness of g^y, where y *in the exponent* is the output of a PRF. They give two constructions (based on DDH and Paillier encryption) in the random oracle model, and discuss multiple applications. For example, simulatable distributed protocols (e.g., distributed key generation or multiparty signing) with fewer rounds of communication (after initially exchanging eVRF verification keys), or hierarchical deterministic key derivation for cryptocurrency wallets.

Observe that our CVRF is an e(C)VRF where $\boldsymbol{y} = G_x(\mathsf{sk})^2 \in \mathbb{Z}_p^n$ and $\mathsf{g}_2^{\boldsymbol{y}} \in \mathbb{G}_2^n$.[10] From a theoretical point of view, our construction improves on Boneh et al. by not relying on random oracles (and also not on a common reference string). In addition, since our construction is also constrainable, our eCVRF enables sub-tree delegation when used for hierarchical key derivations for cryptocurrency wallets (cf. Boneh et al.'s full version for details [BHLS24, §4.6]). When using our construction solely as an eVRF (so no constrained keys are handed out), we conjecture that pseudorandomness can be proven from MQ on its own in the GGM, i.e., MQ+ is not necessary anymore. However, as this text focuses on constrained VRFs, we leave this to future work.

Stake Pooling. Proof of stake blockchains often elect a small committee or slot leader to produce the next block. Each party's probability to be elected is proportional to their stake. Hence, small stakeholders have little incentive to run a computer 24/7 to participate at election. This lowers the resilience of the chain as it concentrates the power in the hands of a few large stakeholders. A mitigation is *stake pooling* where small stakeholders delegate their stake to a pool operator that they trust; the pool operator is elected with probability proportional to the total stake in their pool.

Many blockchain designs use VRFs for such elections (e.g., Algorand [GHM+17] or Ouroboros Praos [DGKR18]). Ignoring details, party i is elected at time t if $F_{\mathsf{sk}_i}(t) > \mathsf{threshold}(\mathsf{stake}_i)$. Since the long-term secret key sk_i must stay secret, delegating stake currently requires on-chain communication, which is expensive. CVRFs enable off-chain delegation: a small stakeholder constrains their key to some time span (i.e., a VRF preimage prefix) and communicates it off-chain to the pool operator, who then acts on behalf of the stakeholder.

Compressible Randomness Beacons. A randomness beacon outputs a pseudorandom value periodically. For an outside observer, the beacon output for time t must be unpredictable at any time $< t$. A compressible randomness beacon enables the beacon operator to provide a concise description of all outputs preceding time t.

This notion was introduced in [HNS25] as one part of a larger protocol for anonymous blockchain payments between light clients. They construct compressible randomness beacons using the GGM-CPRF [GGM86]: The beacon outputs are the leaves of the tree (from left to right as time progresses). Since the GGM-CPRF is not verifiable, a malicious beacon may equivocate and send two different values to different users. In their model, the beacon is trusted (e.g., executed via MPC), so this is not an issue. Using our CVRFs prevents such equivocation attacks without trust assumptions.

Note that generally a malicious beacon operator should not be able to predict outputs in advance as well. This may be achieved by running multiple CVRF-based beacons and relying on non-collusion assumptions between beacons.

[10] Concretely, our construction computes the PRF $\mathsf{PRF} : \{0,1\}^\ell \to \{z^2 | z \in \mathbb{Z}_p\}^n$, $\mathsf{PRF}_{\mathsf{sk}}(x) = G_x(\mathsf{sk})^2 \in \mathbb{Z}_p^n$, in the exponent of its outputs. Since the outputs of PRF are vectors of square numbers, it is simple to distinguish its images from uniformly random vectors over $\mathbb{Z}_p$. However, this problem can simply be fixed by replacing the final squaring step of PRF with applying n uniformly random quadratic polynomials. Security then follows by invoking the normal DDH assumption in the security game. Further, if an application requires eVRFs with a single group element as output, it suffices to simply pick the first output element of our construction, i.e., $\boldsymbol{y}_1$ and $\mathsf{g}_2^{\boldsymbol{y}_1}$.

1.5 Related Work

CPRFs have been introduced by [BW13, KPTZ13, BGI14]. Notably, [BW13] gave instantiations for constraints specified by arbitrary circuits, assuming mmaps. CPRFs for inputs of unbounded length and constraints specified by Turing machines have been given by [AFP16, DKW16] using iO.

The first candidate VRFs have been constructed by [MRV99] assuming the hardness of factoring. Important VRFs have been put forth by Lysyanskaya [Lys02] and Dodis & Yampolskiy [DY05], which described two styles of pairing-based VRFs. Most pairing-based VRFs use q-type assumptions [Lys02, DY05, ACF09, BCKL09, BMR10, HW10, Jag15, Kat17, Yam17, Nie21] to achieve small proof sizes, or are based on DLIN [HJ16, Ros18, Koh19]. [BHKÜ22] showed that it is hard for a pairing-based VRF to achieve simultaneously proofs and assumptions of constant size. It is possible to construct VRFs generically from NIWIs [Bit17, GHKW17]

Initially, [Fuc14, CRV14] constructed CVRFs based on the CPRFs of [BW13]. They achieve bit-fixing and circuit constraints using mmaps. Subsequent CVRF candidates also enjoy circuit constrainability and use iO: [LLC15] achieves selective security, [LZW19] extends their security notion and [ZLX23] achieves adaptive security under iO. Additionally, [DDM17, DDM19] construct CVRFs for unbounded inputs. An exception is [Dat20], which constructs CVRFs from functional encryption (which is equivalent to iO up to subexponential loss). In comparison, our construction only supports less expressive constraints, but builds upon significantly simpler building blocks.

Acknowledgments. We thank Jonas Steinbach and Gertjan De Mulder for helpful discussions on BIP 32. We thank Dennis Hofheinz and Julia Kastner for helpful discussions on early prototypes of our CVRF.

This research was funded in whole or in part by the Austrian Science Fund (FWF) 10.55776/F85. For open access purposes, the author has applied a CC BY public copyright license to any author-accepted manuscript version arising from this submission.

2 Preliminaries

Notation. Denote by $\lambda \in \mathbb{N}$ the security parameter. We denote the output of deterministic and probabilistic algorithms by $y = D(x)$ and $y \leftarrow P(x)$, respectively. Denote by $x \leftarrow\!\!\$\ X$ that x is sampled uniformly at random from the set X. Denote by $2^X = \{A : A \subseteq X\}$ the power set of X. Set $[n] := \{x \in \mathbb{N} : 1 \leq x \leq n\}$ where $\mathbb{N} = \{1, 2, 3, \ldots\}$ is the set of natural numbers.

We denote elements of cyclic groups $\mathbb{G}, \mathbb{G}_1, \mathbb{G}_2$ of order p by monospaced letters $\mathtt{g}, \mathtt{g}_1, \mathtt{g}_2$. Given a vector $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{Z}_p^n$ and a group element $\mathtt{h} \in \mathbb{G}$, we denote by $\mathtt{h}^{\boldsymbol{x}}$ the vector of group elements $(\mathtt{h}^{x_1}, \ldots, \mathtt{h}^{x_n}) \in \mathbb{G}^n$.

For a bit $b \in \{0,1\}$, set $\overline{b} = 1 - b$. We denote the length of a bit string $x \in \{0,1\}^*$ by $|x|$. Let ϵ be the empty word of length 0 and note that $\{0,1\}^0 = \{\epsilon\}$. For $\ell \in \mathbb{N}$, we write $\{0,1\}^\ell$ to denote the bit strings of length ℓ. By $\{0,1\}^{\leq \ell}$ we denote the set of all bit strings of length at most ℓ, including ϵ. We denote string concatenation by $x \cdot y$ and sometimes write xy to avoid clutter. Given a bit string $x \in \{0,1\}^\ell$, we denote its bits by

$x = x_1 \cdots x_\ell$. A bit string $x \in \{0,1\}^*$ is a *prefix* of a bit string $y \in \{0,1\}^*$ if there exists $w \in \{0,1\}^*$ such that $x \cdot w = y$.

On Groups. We will consider a definition of symmetric bilinear groups akin to certified bilinear groups as given in [HJ16].

Definition 1 (Symmetric Bilinear Groups)**.** A (symmetric) *bilinear group*, is a pair of cyclic groups $\mathbb{G}_1, \mathbb{G}_2$ of prime order p with generators g_1, g_2 and a map $e\colon \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ s.t. the following holds: e is bilinear, i.e., $e(\mathsf{g}_1^a, \mathsf{g}_1^b) = e(\mathsf{g}_1, \mathsf{g}_1)^{ab}$ for all $a, b \in \mathbb{Z}$, and e is perfect, i.e., $e(\mathsf{g}_1, \mathsf{g}_1) = \mathsf{g}_2$. We require that e and the group operations on $\mathbb{G}_1$ and $\mathbb{G}_2$ are efficiently computable.

As a small technicality, we additionally require $\mathbb{G}_1, \mathbb{G}_2$ to have unique, recognizable representations of its elements. That is for each element $\mathtt{h}_1 \in \mathbb{G}_1$, there is a canonical encoding as a bit string $\langle \mathtt{h}_1 \rangle_1$. Further, there is a verification algorithm $\mathsf{GrpVfy}_1(s)$, which on input of a supposed bit-string description of an element s outputs 0 or 1. The encoding is uniquely recognizable in the sense that $\mathsf{GrpVfy}_1(s) = 1$ implies that there is $a \in \mathbb{Z}_p$ such that $s = \langle \mathsf{g}_1^a \rangle_1$. The same holds respectively for an encoding $\langle \mathtt{h}_2 \rangle_2$ and a verification algorithm $\mathsf{GrpVfy}_2(s)$ in $\mathbb{G}_2$. When referring to a group element $\mathtt{h}$ in $\mathbb{G}_1$ or $\mathbb{G}_2$ as input in the following, we will implicitly mean its unique encoding, which is implicitly checked to be a valid group element encoding at the beginning of each algorithm.

In this work, we use Maurer's generic model [Mau05] for bilinear groups. Concretely, access to the elements and operations on the two groups $\mathbb{G}_1, \mathbb{G}_2$ (isomorphic to $\mathbb{Z}_p$) with a non-trivial bilinear (symmetric) pairing $e\colon \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ are hidden behind oracles. Further, instead of real group elements, a generic adversary is given handles which only point to elements used by the oracles to perform group operations. Whenever a group query is made to an oracle to perform a certain group operation, handles for the involved group elements and possibly a number in $\mathbb{Z}_p$ has to be supplied. In Maurer's original model, the oracle will then respond with a handle for a new group element (which will be stored internally) and a list of pairs of handles which point to equal group elements.

We will deviate here from Maurer's original model by making equality checks explicit. First note that the adversaries in our setting do not need to output group elements. Instead, they only need to output bits. Hence, we also do not need to give them handles for new group elements after performing group operations. To make this precise, we will define generic adversaries as algorithms that receive handles for group elements as inputs and may only query if some degree-2 polynomial over the exponents of the group elements vanishes. This means, the generic adversary can only query if some computation on its group elements yields the trivial group element, or if two of its computed group elements collide. We model this by allowing it to submit degree-2 polynomials to some oracle, which tells the adversary if the polynomial would vanish on the exponents of the group elements.

The difference between Maurer's original model and our interpretation of it is polynomial: in Maurer's original model, an adversary needs to query q group operations to receive the result of $(q+n)^2$ equality checks, where n is the number of received group elements at the start of the experiment. In our variation, the adversary needs to perform $(q+n)^2$ equality checks, which is by a polynomial amount larger than q.

Definition 2 (Generic Adversaries)**.** An adversary is called *generic* if it works as follows: At the beginning of a security game, the generic adversary receives a number of source group element handles, let's say $\mathsf{g}_1^{x_1}, \ldots, \mathsf{g}_1^{x_n}$ and $\mathsf{g}_2^{y_1}, \ldots, \mathsf{g}_2^{y_m}$, where g_1 and g_2 are fixed generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, with $e(\mathsf{g}_1, \mathsf{g}_1) = \mathsf{g}_2$. It may now only query equality checks, which are modelled by $h \in \mathbb{Z}_p[X_1, \ldots, X_n, Y_1, \ldots, Y_m]$, which must have the shape

$$h(X, Y) = \sum_{i,j=1}^{n} \alpha_{i,j} X_i X_j + \sum_{k=1}^{m} \beta_k Y_k$$

for some $\alpha_{i,j}, \beta_k \in \mathbb{Z}_p$. Whenever the simulator of the group receives such a polynomial h, it has to reply with "$h(\boldsymbol{x}, \boldsymbol{y}) = 0$" if $h(x_1, \ldots, x_n, y_1, \ldots, y_m) = 0$. (This means, the equality check passes.) Otherwise, the simulator has to reply with "$h(\boldsymbol{x}, \boldsymbol{y}) \neq 0$".

On Pseudorandom Functions.

Definition 3 (Constrained Pseudorandom Function)**.** A *constrained pseudorandom function* (CPRF) with input domain $\mathcal{X} = \mathcal{X}(\lambda)$, output codomain $\mathcal{Y} = \mathcal{Y}(\lambda)$, and set of efficiently representable constraints $\mathcal{C} = \mathcal{C}(\lambda) \subseteq 2^{\mathcal{X}}$ where $\emptyset \in \mathcal{C}$ is a tuple $\mathsf{CPRF} = (\mathsf{Setup}, \mathsf{Constrain}, \mathsf{Eval})$ of three (probabilistic) polynomial-time algorithms

$\mathsf{Setup}(1^\lambda) \to \mathsf{sk}_\emptyset$**:** Given a unary encoded security parameter, samples a secret key $\mathsf{sk}_\emptyset$, which may evaluate all inputs $\mathcal{X}$.

$\mathsf{Constrain}(\mathsf{sk}_C, C') = \mathsf{sk}_{C'}$**:** Given a secret key sk_C with constraint $C \in \mathcal{C}$ and a constraint $C' \in \mathcal{C}$, outputs the constrained secret key $\mathsf{sk}_{C'}$ or $\perp$.

$\mathsf{Eval}(\mathsf{sk}_C, x) = y$**:** Given a secret key sk_C with constraint $C \in \mathcal{C}$ and an input $x \in \mathcal{X}$, outputs $y \in \mathcal{Y}$ or $\perp$.

A CPRF has to fulfill the following properties:

d**-Delegability:** For any constraints $\emptyset = C_0 \subseteq \cdots \subseteq C_d$ with $C_i \in \mathcal{C}$, any $x \in \mathcal{X} \setminus C_d$ and any $\mathsf{sk}_{C_0} \leftarrow \mathsf{Setup}(1^\lambda)$ it holds that

$$\mathsf{Eval}(\mathsf{sk}_{C_0}, x) = \cdots = \mathsf{Eval}(\mathsf{sk}_{C_d}, x)$$

where $\mathsf{sk}_{C_1} = \mathsf{Constrain}(\mathsf{sk}_{C_0}, C_1)$, ..., and $\mathsf{sk}_{C_d} = \mathsf{Constrain}(\mathsf{sk}_{C_{d-1}}, C_d)$.

(t, q, μ)**-Selective Pseudorandomness:** Consider the following game parameterized by λ that is played by an adversary $\mathcal{A}$ making at most q queries:

1. Sample $\mathsf{sk}_\emptyset \leftarrow \mathsf{Setup}(1^\lambda)$ and $b \leftarrow\!\!\$\ \{0, 1\}$.
2. $\mathcal{A}$ outputs a challenge point $x^* \in \mathcal{X}$.
3. $\mathcal{A}$ outputs a list of constraint queries $(C_i)_{i \in [q_1]}$ and a list of input queries $(x_j)_{j \in [q_2]}$ where $q \geq q_1 + q_2$.
4. $\mathcal{A}$ receives $\mathsf{sk}_{C_i} \leftarrow \mathsf{Constrain}(\mathsf{sk}_\emptyset, C_i)$ for $i \in [q_1]$ and $y_j = \mathsf{Eval}(\mathsf{sk}_\emptyset, x_j)$ for $j \in [q_2]$.
5. $\mathcal{A}$ receives y_b^* where $y_0^* = \mathsf{Eval}(\mathsf{sk}_\emptyset, x^*)$ and $y_1^* \leftarrow\!\!\$\ \mathcal{Y}$.

6. $\mathcal{A}$ outputs a guess b^*.

We say "$\mathcal{A}$ wins" if and only if $b = b^*$, $x^* \in C_i$ for $i \in [q_1]$, and $x_j \neq x^*$ for $j \in [q_2]$. The *advantage* of $\mathcal{A}$ in the above game is

$$\mathrm{Adv}^{\mathcal{A}}_{\mathsf{CPRF}}(\lambda) = |\Pr[\mathcal{A} \text{ wins}] - 1/2|$$

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

CPRF is (t, q, μ)-*selective pseudorandom* if, for every $\mathcal{A}$ running in time at most $t(\lambda)$ making at most $q(\lambda)$ queries, it holds that $\mathrm{Adv}^{\mathcal{A}}_{\mathsf{CPRF}}(\lambda) \leq \mu(\lambda)$.

We call CPRF *delegable* if it is d-delegable for every $d \in \mathrm{poly}(\lambda)$, and (t, μ)-*selective pseudorandom* if it is (t, q, μ)-*selective pseudorandom* for every $q \in \mathrm{poly}(\lambda)$.

Definition 4 (Puncturable CPRF)**.** A CPRF with input domain $\mathcal{X}$ and set of constraints $\mathcal{C}$ is *puncturable* if $\{x\} \in \mathcal{C}$ for every $x \in \mathcal{X}$.

Definition 5 (Constrained Verifiable Pseudorandom Function)**.** A *constrained verifiable pseudorandom function* (CVRF) with public parameter space $\mathcal{P} = \mathcal{P}(\lambda)$, input domain $\mathcal{X} = \mathcal{X}(\lambda)$, output codomain $\mathcal{Y} = \mathcal{Y}(\lambda)$, and set of constraints $\mathcal{C} = \mathcal{C}(\lambda) \subseteq 2^{\mathcal{X}}$ where $\emptyset \in \mathcal{C}$ is a tuple $\mathsf{CVRF} = (\mathsf{Setup}, \mathsf{Constrain}, \mathsf{Eval}, \mathsf{Verify}, \mathsf{VerifyC})$ of five (probabilistic) polynomial-time algorithms

$\mathsf{Setup}(1^\lambda) \to (\mathsf{pp}, \mathsf{sk}_\emptyset, \mathsf{vk}, \pi_\emptyset)$**:** Given a unary encoded security parameter, samples public parameters $\mathsf{pp} \in \mathcal{P}$, secret key $\mathsf{sk}_\emptyset$, verification key vk, and proof $\pi_\emptyset$.[11]

$\mathsf{Constrain}_{\mathsf{pp}}(\mathsf{sk}_C, \pi_C, C') \to (\mathsf{sk}_{C'}, \pi_{C'})$**:** Given a secret key sk_C with constraint $C \in \mathcal{C}$, proof π_C, and a constraint $C' \in \mathcal{C}$, outputs the constrained secret key $\mathsf{sk}_{C'}$ and proof $\pi_{C'}$, or $\perp$.

$\mathsf{Eval}_{\mathsf{pp}}(\mathsf{sk}_C, \pi_C, x) \to (y, \pi)$**:** Given a secret key sk_C with constraint $C \in \mathcal{C}$ and an input $x \in \mathcal{X}$, outputs $y \in \mathcal{Y}$ and proof π, or $\perp$.

$\mathsf{VerifyC}_{\mathsf{pp}}(\mathsf{vk}, C, \mathsf{sk}_C, \pi_C) = b$**:** Given the verification key vk, constraint $C \in \mathcal{C}$, constrained secret key sk_C, and proof π_C, outputs $b \in \{\mathsf{ACCEPT}, \mathsf{REJECT}\}$.

$\mathsf{Verify}_{\mathsf{pp}}(\mathsf{vk}, x, y, \pi) = b$**:** Given the verification key vk, input $x \in \mathcal{X}$, output $y \in \mathcal{Y}$, and proof π, outputs $b \in \{\mathsf{ACCEPT}, \mathsf{REJECT}\}$.

A CVRF has to fulfill the following properties:

d**-Delegability:** For any constraints $\emptyset = C_0 \subseteq \cdots \subseteq C_d$ with $C_i \in \mathcal{C}$, and $x \in \mathcal{X} \setminus C_d$ it holds that

$$\Pr\left[\begin{array}{l} \mathsf{VerifyC}_{\mathsf{pp}}(\mathsf{vk}, C_0, \mathsf{sk}_{C_0}, \pi_{C_0}) = \cdots = \mathsf{VerifyC}_{\mathsf{pp}}(\mathsf{vk}, C_d, \mathsf{sk}_{C_d}, \pi_{C_d}) = \\ \mathsf{Verify}_{\mathsf{pp}}(\mathsf{vk}, x, y_0, \pi_0) = \cdots = \mathsf{Verify}_{\mathsf{pp}}(\mathsf{vk}, x, y_d, \pi_d) = \mathsf{ACCEPT} \end{array}\right] = 1$$

where the probability is taken over $(\mathsf{pp}, \mathsf{sk}_{C_0}, \mathsf{vk}, \pi_{C_0}) \leftarrow \mathsf{Setup}(1^\lambda)$, and $(\mathsf{sk}_{C_i}, \pi_{C_i}) \leftarrow \mathsf{Constrain}(\mathsf{sk}_{C_{i-1}}, \pi_{C_{i-1}}, C_i)$ and $(y_i, \pi_i) \leftarrow \mathsf{Eval}_{\mathsf{pp}}(\mathsf{sk}_{C_i}, \pi_{C_i}, x)$ for $i \in [d]$.

[11] The proof $\pi_\emptyset$ is a formality that allows us to streamline definitions and syntax. $\pi_\emptyset$ can be seen as part of $\mathsf{sk}_\emptyset$.

Unique Provability: For all (even malformed) public parameters pp, verification keys vk, and proofs π, π' as well as all inputs $x \in \mathcal{X}$ and outputs $y, y' \in \mathcal{Y}$, it holds that

$$\mathsf{Verify}_{\mathsf{pp}}(\mathsf{vk}, x, y, \pi) = \mathsf{Verify}_{\mathsf{pp}}(\mathsf{vk}, x, y', \pi') = \mathsf{ACCEPT} \implies y = y'.$$

Functionality Binding : For all (even malformed) public parameters pp, verification keys vk, secret keys $\mathsf{sk}_C, \mathsf{sk}'_C$ and proofs π_C, π'_C the following holds: If

$$\mathsf{VerifyC}_{\mathsf{pp}}(\mathsf{vk}, C, \mathsf{sk}_C, \pi_C) = \mathsf{VerifyC}_{\mathsf{pp}}(\mathsf{vk}, C, \mathsf{sk}'_C, \pi'_C) = \mathsf{ACCEPT},$$

then, for all inputs $x \in \mathcal{X} \setminus C$, for all $(y, \pi) \leftarrow \mathsf{Eval}(\mathsf{sk}_C, \pi_C, x)$ and $(y', \pi') \leftarrow \mathsf{Eval}(\mathsf{sk}'_C, \pi'_C, x)$ it holds that $y = y'$.

(t, q, μ)**-Selective Pseudorandomness:** Consider the following game parameterized by λ that is played by an adversary $\mathcal{A}$ making at most q queries:

1. Sample $(\mathsf{pp}, \mathsf{sk}_\emptyset, \mathsf{vk}, \pi_\emptyset) \leftarrow \mathsf{Setup}(1^\lambda)$ and $b \leftarrow\!\!\$\ \{0, 1\}$.
2. $\mathcal{A}$ outputs a challenge point $x^* \in \mathcal{X}$.
3. $\mathcal{A}$ gets pp and vk.
4. $\mathcal{A}$ outputs a list of constraint queries $(C_i)_{i \in [q_1]}$ and a list of input queries $(x_j)_{j \in [q_2]}$ where $q \geq q_1 + q_2$.
5. $\mathcal{A}$ receives $(\mathsf{sk}_{C_i}, \pi_{C_i}) \leftarrow \mathsf{Constrain}(\mathsf{sk}_\emptyset, \pi_\emptyset, C_i)$ for $i \in [q_1]$ and $(y_j, \pi_j) \leftarrow \mathsf{Eval}(\mathsf{sk}_\emptyset, \pi_\emptyset, x_j)$ for $j \in [q_2]$.
6. $\mathcal{A}$ receives y_b^* where $(y_0^*, \pi_0^*) = \mathsf{Eval}(\mathsf{sk}_\emptyset, x^*)$ and $y_1^* \leftarrow\!\!\$\ \mathcal{X}$.
7. $\mathcal{A}$ outputs a guess b^*.

We say "$\mathcal{A}$ wins" if and only if $b = b^*$, $x^* \in C_i$ for $i \in [q_1]$, and $x_j \neq x^*$ for $j \in [q_2]$. The *advantage* of $\mathcal{A}$ in the above game is

$$\mathrm{Adv}^{\mathcal{A}}_{\mathsf{CVRF}}(\lambda) = |\Pr[\mathcal{A} \text{ wins}] - 1/2|$$

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

CVRF is (t, q, μ)*-selective pseudorandom* if, for every $\mathcal{A}$ running in time at most $t(\lambda)$ making at most $q(\lambda)$ queries, it holds that $\mathrm{Adv}^{\mathcal{A}}_{\mathsf{CVRF}}(\lambda) \leq \mu(\lambda)$.

We call CVRF *delegable* if it is d-delegable for every $d \in \mathrm{poly}(\lambda)$, and (t, μ)*-selective pseudorandom* if it is (t, q, μ)*-selective pseudorandom* for every $q \in \mathrm{poly}(\lambda)$.

Definition 6 (Constraing-Hiding)**.** CVRF is *constraint-hiding* if we have for all constraints $C_1 \subseteq C_2$ and $x \in \mathcal{X} \setminus C_1$ the following equalities of distributions:

$$\mathsf{Constrain}_{\mathsf{pp}}(\mathsf{sk}_\emptyset, \pi_\emptyset, C_2) \equiv \mathsf{Constrain}_{\mathsf{pp}}(\mathsf{sk}_{C_1}, \pi_{C_1}, C_2),$$
$$\mathsf{Eval}_{\mathsf{pp}}(\mathsf{sk}_\emptyset, \pi_\emptyset, x) \equiv \mathsf{Eval}_{\mathsf{pp}}(\mathsf{sk}_{C_1}, \pi_{C_1}, x)$$

where we sample $(\mathsf{pp}, \mathsf{sk}_\emptyset, \mathsf{vk}, \pi_\emptyset) \leftarrow \mathsf{Setup}(1^\lambda)$ and $(\mathsf{sk}_{C_1}, \pi_{C_1}) \leftarrow \mathsf{Constrain}(\mathsf{sk}_\emptyset, \pi_\emptyset, C_1)$.

Remark 1. *Note that our notion of constraint-hiding (Def. 6) is slightly stricter than the notion of Fuchsbauer [Fuc14]. Also, note that the selective pseudorandomness game in Def. 5 does not cover complex cases of repeated or nested constraining a passive adversary may observe in a scenario where multiple parties might receive constrained keys and further constrain them.*

As an example, consider the following case that is not covered: Let x^* be the point challenged by the adversary. Additionally, it asks the challenger to generate a constraint key sk_C for a constraint with $x^* \notin C$ and then further constrain sk_C to $\mathsf{sk}_{C'}$ with $x^* \in C' \supseteq C$. Finally, we hand $\mathsf{sk}_{C'}$ to the adversary. Creating $\mathsf{sk}_{C'}$ and its corresponding proof $\pi_{C'}$ in this way results in a potentially different output than constraining $\mathsf{sk}_\emptyset$ at C' directly. Such types of constraining are not covered by the security notion of 5.

The constraint-hiding property solves this problem by ensuring that the proofs of constrained keys and output values are independent of the (constrained) key they were computed from. The construction we give in § 3 is constraint-hiding in the strongest possible way, as all outputs, proofs and constrained keys of it are determined deterministically by its verification key.

Definition 7 (Prefix-Constrained CPRFs and CVRFs)**.** A CPRF or CVRF with input domain $\mathcal{X} = \{0,1\}^\ell$ is *prefix-constrained* if, for every $x \in \{0,1\}^{\leq \ell}$,

$$\{y \in \{0,1\}^\ell \mid x \text{ is not a prefix of } y\} \in \mathcal{C}.$$

For $C = \{y \in \{0,1\}^\ell \mid x \text{ is not a prefix of } y\}$, we will usually denote the outputs of $\mathsf{Constrain}(\mathsf{sk}_\emptyset, \pi_\emptyset, C)$ by (sk_x, π_x) instead of (sk_C, π_C).

On Quadratic Polynomials. We will introduce here some notation on polynomials of degree 2 that will span through this work.

Definition 8. Let $n, p \in \mathbb{N}$. A **degree-2** map $F : \mathbb{Z}_p^n \longrightarrow \mathbb{Z}_p^n$ is a map whose outputs are computed by n polynomials $f_1, \ldots, f_n \in \mathbb{Z}_p[X_1, \ldots, X_n]$ of degree ≤ 2. Denote the set of all degree-2 maps by

$$Q_{p,n} := \{F : \mathbb{Z}_p^n \to \mathbb{Z}_p^n \mid F \text{ is a degree-2 map}\}.$$

In this work, we will sometimes refer to $Q_{p,n}$ as Q, whenever p, n are clear from context.

Definition 9. For $\ell \in \mathbb{N}$, denote by $Q_{p,n}^{\ell \times 2}$ the set of lists of ℓ pairs of functions in $Q_{p,n}$. We will denote an element $\mathcal{G} \in Q_{p,n}^{\ell \times 2}$ by

$$\mathcal{G} = ((G_0^{(1)}, G_1^{(1)}), \ldots, (G_0^{(\ell)}, G_1^{(\ell)}))$$

where $G_0^{(1)}, \ldots, G_0^{(\ell)}, G_1^{(1)}, \ldots, G_1^{(\ell)} \in Q_{p,n}$ are all degree-2 maps from $\mathbb{Z}_p^n$ to $\mathbb{Z}_p^n$. Further, for a bit string $x = x_1 \cdots x_{\ell'} \in \{0,1\}^{\leq \ell}$, we set

$$G_x := G_{x_{\ell'}}^{(\ell')} \circ \cdots \circ G_{x_1}^{(1)} : \mathbb{Z}_p^n \to \mathbb{Z}_p^n.$$

For the sake of completeness, let G_ϵ denote the identity function on $\mathbb{Z}_p^n$.

Finally, let $e\colon \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ be a bilinear pairing with generators $\mathtt{g}_1 \in \mathbb{G}_1$, $\mathtt{g}_2 \in \mathbb{G}_2$. For a degree-2 map $F \in Q$, we define an efficiently computable function $e_F : \mathbb{G}_1^n \to \mathbb{G}_2^n$ as follows: let $f_1, \ldots, f_n \in \mathbb{Z}_p[X_1, \ldots, X_n]$ be the polynomials that compute the outputs of F. Decompose each f_i as

$$f_i = \sum_{1 \le j \le k \le n} c_{j,k}^{(i)} \cdot X_j X_k + \sum_{j \in [n]} c_{0,j}^{(i)} \cdot X_j + c_{0,0}^{(i)}$$

with coefficients $c_{j,k}^{(i)} \in \mathbb{Z}_p$. Then, the output of e_F on input $\mathtt{g}_1^{\boldsymbol{s}}$ is given by $e_F(\mathtt{h}_1, \ldots, \mathtt{h}_n) = (\mathtt{h}'_1, \ldots, \mathtt{h}'_n)$ where

$$\mathtt{h}'_i := \left(\prod_{1 \le j \le k \le n} e(\mathtt{h}_j, \mathtt{h}_k)^{c_{j,k}^{(i)}} \right) \cdot \left(\prod_{j \in [n]} e(\mathtt{h}_j, \mathtt{g}_1)^{c_{0,j}^{(i)}} \right) \cdot \mathtt{g}_2^{c_{0,0}^{(i)}}.$$

Note that we have $e_F(\mathtt{g}_1^{\boldsymbol{s}}) = \mathtt{g}_2^{F(\boldsymbol{s})}$ for $\boldsymbol{s} \in \mathbb{Z}_p^n$.

3 Our CVRF Construction

Our construction draws inspiration from the CPRF of Goldreich, Goldwasser and Micali [GGM86] (GGM-CPRF). The GGM-CPRF's input domain is $\{0,1\}^\ell$ and, conceptually, it may be viewed as a binary tree of depth ℓ. Each node is identified by a bit string $x \in \{0,1\}^{\le \ell}$ where the root node is ϵ, and the children of a node x are $x0$ and $x1$ respectively. The set of leafs is $\{0,1\}^\ell$. Additionally, each node has a label associated with it, and these labels are defined inductively by a length-doubling PRG $G = (G_0, G_1)$. The label of the root node is the CPRF's root secret keys sk_ϵ. The label of others node is computed recursively: the label of the left and right child is yielded by applying G_0 and G_1 to the parent's label, respectively. In the GGM-CPRF, constrained keys correspond to the labels of inner nodes, while outputs correspond to the labels of leaf nodes.

Our CVRF construction (denoted by CVRF) shares the input domain $\{0,1\}^\ell$ and PRG-defined tree structure with the GGM-CPRF. To enable verification, however, our construction differs in several aspects:

1. Our PRGs are degree-2 polynomials over $\mathbb{Z}_p$ in n variables (as explained in Def. 8). Additionally, for each level of the tree, we use a *fresh* PRG $G^{(i)} \leftarrow\!\!\$\ Q^2$ that is sampled uniformly and independently of all other PRGs.

2. Due to our choice of PRGs, each node's label is a value in $\mathbb{Z}_p^n$. In particular, the root secret key sk_ϵ is a uniformly random vector $\boldsymbol{s} \leftarrow\!\!\$\ \mathbb{Z}_p^n$.

3. While constrained keys are the labels of inner nodes as in the GGM-CPRF, outputs are not the labels of leaf nodes directly, but a function of them. In particular, outputs are vectors of group elements in $\mathbb{G}_2^n$ where, on input $x \in \{0,1\}^\ell$, the output is $\mathtt{g}_2^{G_x(\boldsymbol{s})^2}$ (where $G_x(\boldsymbol{s})$ is the label of the leaf node x by the notational shorthand introduced in Def. 9, and $G_x(\boldsymbol{s})^2$ is the vector of squares of entries of $G_x(\boldsymbol{s})$).

4. To enable verification of constrained keys and outputs, nodes not only have a label, but also a proof element associated to them.

(a) The verification key is $\mathsf{vk} = \mathsf{g}_2^{\mathsf{sk}_\epsilon} = \mathsf{g}_2^{\boldsymbol{s}}$.

(b) Each node $x \in \{0,1\}^{\leq \ell}$ has the proof element $\mathsf{g}_1^{G_x(\boldsymbol{s})}$ associated to it.

5. Each constrained key or output is accompanied by the proof elements along the path from the root node to itself. Since the labels are inductively defined via degree-2 polynomials, the proof elements enable efficient verification by using the pairing e.

Given this high-level overview, we give the pseudocode of CVRF in Fig. 2. To avoid redundant code, CVRF uses internally the helper algorithms HelperEval and HelperVerify, which are called whenever outputs and constrained keys are generated and verified, respectively.

Theorem 3. CVRF *fulfills* Delegability, Unique Provability *and* Functionality Binding *(Def. 5). In addition, it is* constraint-hiding *(Def. 6).*

Proof. Delegability follows almost directly from the fact that HelperVerify accepts outputs of HelperEval by construction and the fact that HelperEval and HelperVerify do not differentiate between root and constrained secret keys. *Constraint-Hiding* also follows by construction since HelperEval is deterministic.

With respect to *Unique Provability* and *Functionality Binding*, we will show that, for any input $x \in \{0,1\}^{\leq \ell}$, and any verification key $\mathsf{vk} \in \mathbb{G}_2^n$ there is a unique input $(\pi_x \in (\mathbb{G}_1^n)^{\ell'}, \tilde{\pi}^{(\ell')} \in \mathbb{G}_1^n)$ resulting in $\mathsf{HelperVerify}(\mathsf{vk}, x, \pi_x, \tilde{\pi}^{(\ell')}) = \mathsf{ACCEPT}$.

This directly implies the unique provability as Verify only accepts an input $\boldsymbol{y}$ if it fulfills $\boldsymbol{y} = e(\tilde{\pi}^{(\ell')}, \tilde{\pi}^{(\ell')})$ for a thus unique $\tilde{\pi}^{(\ell')}$ predetermined by HelperVerify. Regarding functionality binding, we can actually show the stronger notion, that for any constraint $x \in \{0,1\}^{<\ell}$ constrained keys $\mathsf{sk}_x, \mathsf{sk}'_x$ and proofs π_x, π'_x, $\mathsf{VerifyC}_{\mathsf{pp}}(\mathsf{vk}, x, \mathsf{sk}_x, \pi_x) = \mathsf{VerifyC}_{\mathsf{pp}}(\mathsf{vk}, x, \mathsf{sk}'_x, \pi'_x) = \mathsf{ACCEPT}$ already implies that $\mathsf{sk}_x = \mathsf{sk}'_x$. This follows in the same way as unique provability, as VerifyC only accepts a key sk_x if it conforms to $\mathsf{g}_1^{\mathsf{sk}_x} = \tilde{\pi}^{(\ell')}$ for this unique value $\tilde{\pi}^{(\ell')}$.

Let us now show that there is one unique accepting input to HelperVerify. Consider any $\mathsf{pp} = ((G_0^{(i)}, G_1^{(i)})_{i=1}^{\ell})$ and any $\mathsf{vk} \in \mathbb{G}_2^n$. Let $\ell' \leq \ell$ and $x \in \{0,1\}^{\ell'}$. Suppose we are given two proofs $\pi_1 = (\tilde{\pi}_1^{(i)})_{i=0}^{\ell'-1}$ and $\pi_2 = (\tilde{\pi}_2^{(i)})_{i=0}^{\ell'-1} \in (\mathbb{G}_1^n)^{\ell'}$ with corresponding inputs $\tilde{\pi}_1^{(\ell')}$ and $\tilde{\pi}_2^{(\ell')}$ that both result in ACCEPT (we use the subscripts 1 and 2 to differentiate between the two executions of HelperVerify). We will show by induction on $0 \leq i \leq \ell'$ that $\tilde{\pi}_1^{(i)} = \tilde{\pi}_2^{(i)} = \mathsf{g}_1^{G_{x_1 \ldots x_i}(\mathsf{sk}_\epsilon)}$ for every i.

For the base case $i = 0$ it holds that $\tilde{\pi}_1^{(0)} = \tilde{\pi}_2^{(0)} = \mathsf{g}_1^{\mathsf{sk}_\epsilon} = \mathsf{g}_1^{G_\epsilon(\mathsf{sk}_\epsilon)}$ as otherwise the comparison with vk in Line 2 would not pass. For the induction step assume that $\tilde{\pi}_1^{(i-1)} = \tilde{\pi}_2^{(i-1)} = \mathsf{g}_1^{G_{x_1 \ldots x_{i-1}}(\mathsf{sk}_\epsilon)}$ holds. By the check in Line 4, $G_{x_i}^{(i)}$ is a degree-2 map, so the output of the pairing e in Line 5 is well-defined. The check in Line 5 ensures that $\tilde{\pi}_1^{(i)} = \tilde{\pi}_2^{(i)} = \mathsf{g}_1^{G_{x_1 \ldots x_i}(\mathsf{sk}_\epsilon)}$, completing the proof. □

4 Security Proof by Recursive Assumptions

4.1 Diffie-Hellman Assumptions

The security of our efficient construction in § 3 is based on two assumptions. The first one assumes the hardness of the Decisional Square Diffie-Hellman (DSDH) problem over

$\mathsf{CVRF.Setup}(1^\lambda)$:

01 Sample $\boldsymbol{s} \leftarrow\!\!\$\ \mathbb{Z}_p^n$; Set $\mathsf{sk}_\epsilon = \boldsymbol{s}$
02 Set $\mathsf{vk} = \mathsf{g}_2^{\boldsymbol{s}}$
03 Sample degree-2 maps $\mathcal{G} \leftarrow\!\!\$\ Q_{p,n}^{\ell\times 2}$; Set $\mathsf{pp} = \mathcal{G}$
04 Output $(\mathsf{pp}, \mathsf{sk}_\epsilon, \mathsf{vk}, \pi_\epsilon = \emptyset)$

$\mathsf{HelperEval}_{\mathsf{pp}}(\mathsf{sk}_{x'} \in \mathbb{Z}_p^n, \pi_{x'} \in (\mathbb{G}_1^n)^{\ell'}, x \in \{0,1\}^{\ell''})$:

01 Check that $|x'| = \ell' \le \ell'' \le \ell$; otherwise, abort
02 Parse $\pi_{x'} = (\tilde{\pi}^{(i)})_{i=0}^{\ell'-1}$, $\mathsf{pp} = ((G_0^{(i)}, G_1^{(i)}))_{i=1}^{\ell}$, and $\boldsymbol{s}_{\ell'} = \mathsf{sk}_{x'}$
03 For $i \in \{\ell', \ldots, \ell''-1\}$:
04 Set $\tilde{\pi}^{(i)} = \mathsf{g}_1^{\boldsymbol{s}_i}$
05 Set $\boldsymbol{s}_{i+1} = G_{x_i}^{(i)}(\boldsymbol{s}_i)$
06 Output $(\boldsymbol{y} = \boldsymbol{s}_{\ell''}, \pi_x = (\tilde{\pi}^{(i)})_{i=0}^{\ell''-1})$

$\mathsf{CVRF.Constrain}_{\mathsf{pp}}(\mathsf{sk}_{x'} \in \mathbb{Z}_p^n, \pi_{x'} \in (\mathbb{G}_1^n)^{\ell'}, x \in \{0,1\}^{<\ell})$:

01 Check whether x' is a prefix of x and $|x| < \ell$; else, abort
02 Output $(\mathsf{sk}_x, \pi_x) \leftarrow \mathsf{HelperEval}_{\mathsf{pp}}(\mathsf{sk}_{x'}, \pi_{x'}, x)$

$\mathsf{CVRF.Eval}_{\mathsf{pp}}(\mathsf{sk}_{x'} \in \mathbb{Z}_p^n, \pi_{x'} \in (\mathbb{G}_1^n)^{\ell'}, x \in \{0,1\}^{\ell})$:

01 Check whether x' is a prefix of x and $|x| = \ell$; else, abort
02 $(\hat{\boldsymbol{y}}, \pi_x) \leftarrow \mathsf{HelperEval}_{\mathsf{pp}}(\mathsf{sk}_{x'}, \pi_{x'}, x)$
03 Output $(\boldsymbol{y} = \mathsf{g}_2^{\hat{\boldsymbol{y}}^2}, \pi = (\pi_x, \mathsf{g}_1^{\hat{\boldsymbol{y}}}))$ where $(\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_n)^2 = (\hat{\boldsymbol{y}}_1^2, \ldots, \hat{\boldsymbol{y}}_n^2) \in \mathbb{Z}_p^n$

$\mathsf{HelperVerify}_{\mathsf{pp}}(\mathsf{vk} \in \mathbb{G}_2^n, x \in \{0,1\}^{\ell'}, \pi_x \in (\mathbb{G}_1^n)^{\ell'}, \tilde{\pi}^{(\ell')} \in \mathbb{G}_1^n)$:

01 Parse $\pi_x = (\tilde{\pi}^{(i)})_{i=0}^{\ell'-1}$ and $\mathsf{pp} = ((G_0^{(i)}, G_1^{(i)}))_{i=1}^{\ell}$
02 If $\mathsf{vk} \neq e(\tilde{\pi}^{(0)}, \mathsf{g}_1)$, output REJECT
03 For $1 \le i \le \ell'$:
04 If $G_{x_i}^{(i)}$ is not a degree-2 map, output REJECT
05 If $e(\tilde{\pi}^{(i)}, \mathsf{g}_1) \neq e_{G_{x_i}^{(i)}}(\tilde{\pi}^{(i-1)})$, output REJECT
06 Output ACCEPT

$\mathsf{CVRF.VerifyC}_{\mathsf{pp}}(\mathsf{vk} \in \mathbb{G}_2^n, x \in \{0,1\}^{\ell'}, \mathsf{sk}_x \in \mathbb{Z}_p^n, \pi_x \in (\mathbb{G}_1^n)^{\ell'})$:

01 If $\ell' \ge \ell$, output REJECT
02 Output $\mathsf{HelperVerify}_{\mathsf{pp}}(\mathsf{vk}, x, \pi_x, \mathsf{g}_1^{\mathsf{sk}_x})$

$\mathsf{CVRF.Verify}_{\mathsf{pp}}(\mathsf{vk} \in \mathbb{G}_2^n, x \in \{0,1\}^{\ell}, \boldsymbol{y} \in \mathbb{G}_2^n, \pi = (\pi_x \in (\mathbb{G}_1^n)^{\ell}, \tilde{\pi}^{(\ell)} \in \mathbb{G}_1^n))$:

01 If $\mathsf{HelperVerify}_{\mathsf{pp}}(\mathsf{vk}, x, \pi_x, \tilde{\pi}^{(\ell)}) = \mathsf{REJECT}$, output REJECT
02 If $e(\tilde{\pi}^{(\ell)}, \tilde{\pi}^{(\ell)}) = \boldsymbol{y}$, output ACCEPT; else REJECT

Figure 2: Pseudocode of CVRF.

a cyclic group $\mathbb{G}$ of order p with generator g, which consists in distinguishing a pair of the form $(\mathsf{g}^a, \mathsf{g}^b)$ where $a, b \leftarrow\!\!\$\ \mathbb{Z}_p$ from a pair $(\mathsf{g}^a, \mathsf{g}^{a^2})$ where $a \leftarrow\!\!\$\ \mathbb{Z}_p$.

Definition 10 (DSDH Assumption)**.** Fix $n \in \mathrm{poly}(\lambda)$, a prime $p \leq 2^{\mathrm{poly}(\lambda)}$ and a cyclic group $\mathbb{G}$ of order p with generator $\mathsf{g} \in \mathbb{G}$.

For an adversary $\mathcal{A}$, we define its advantage at solving the *Decisional Square Diffie-Hellman Problem* (DSDH) problem in $\mathbb{G}$ by

$$\mathrm{Adv}^{\mathcal{A}}_{\mathsf{DSDH}}(\lambda) := |\Pr_{\boldsymbol{a},\boldsymbol{b}\leftarrow\!\!\$\mathbb{Z}_p^n}[\mathcal{A}(\mathsf{g}^{\boldsymbol{a}}, \mathsf{g}^{\boldsymbol{b}}) = 1] - \Pr_{\boldsymbol{a}\leftarrow\!\!\$\mathbb{Z}_p^n}[\mathcal{A}(\mathsf{g}^{\boldsymbol{a}}, \mathsf{g}^{\boldsymbol{a}^2}) = 1]|.$$

For non-negative functions $t = t(\lambda), \mu = \mu(\lambda)$, the *Decisional Square Diffie-Hellman assumption* (t, μ)-DSDH states that we have $\mathrm{Adv}^{\mathcal{A}}_{\mathsf{DSDH}} \leq \mu$ for all algorithms $\mathcal{A}$ that run in time at most t.

Note that the DSDH assumption is usually stated for $n = 1$. Since DSDH is rerandomizable, there is a tight reduction of the corresponding problem from the case $n = 1$ to the case $n > 1$.

We now state our novel *Recursive Decisional Diffie-Hellman* assumption. It consists of two challenges: an if-challenge and a then-challenge. The assumption postulates that for every adversary $\mathcal{A}_{\mathsf{then}}$ for the then-challenge, there exists an adversary for the if-challenge of comparable time complexity and advantage. In other words, if the if-challenge is hard, then the then-challenge has to be hard, too. Assumptions of this structure are already known in the lattice-setting as evasive Learning With Errors assumptions [Wee22, Tsa22, BÜW24] and in the pairing setting [AWY20] However, we note that this assumption is just a cornerstone for the security proof of our CVRF, as we prove the assumption sound in the GGM under falsifiable contained assumptions.

Assumption 1 (Recursive DDH)**.** Fix $n, m_1, m_2 \in \mathrm{poly}(\lambda)$, a prime $p \leq 2^{\mathrm{poly}(\lambda)}$ and a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ of groups of order p with generators $\mathsf{g}_1 \in \mathbb{G}_1$, $\mathsf{g}_2 \in \mathbb{G}_2$. Let Samp be an algorithm that on input $1^\lambda, p, n$, outputs descriptions of

1. an efficiently computable function $c : \mathbb{Z}_p^n \to \mathbb{Z}_p^n$,
2. an efficiently computable function $h_0 : \mathbb{Z}_p^n \to \mathbb{Z}_p^{m_0}$,
3. an efficiently computable function $h_1 : \mathbb{Z}_p^n \to \mathbb{Z}_p^{m_1}$,
4. and an efficiently computable function $h_2 : \mathbb{Z}_p^n \to \mathbb{Z}_p^{m_2}$.

For an adversary $\mathcal{A}_{\mathsf{if}}$, we define its advantage in the if-challenge by

$$\begin{aligned}\mathrm{Adv}^{\mathcal{A}_{\mathsf{if}}}_{\mathsf{if}} = |&\Pr[\mathcal{A}_{\mathsf{if}}(c, h_0, h_1, h_2, h_0(\boldsymbol{s}), \mathsf{g}_1^{h_1(\boldsymbol{s})}, \mathsf{g}_2^{h_2(\boldsymbol{s})}, \mathsf{g}_2^{\boldsymbol{s}}, \mathsf{g}_2^{c(\boldsymbol{s})}) = 1] \\ &- \Pr[\mathcal{A}_{\mathsf{if}}(c, h_0, h_1, h_2, h_0(\boldsymbol{s}), \mathsf{g}_1^{h_1(\boldsymbol{s})}, \mathsf{g}_2^{h_2(\boldsymbol{s})}, \mathsf{g}_2^{\boldsymbol{s}}, \mathsf{g}_2^{\boldsymbol{r}}) = 1]|\end{aligned}$$

where the probabilities are taken over $(c, h_0, h_1, h_2) \leftarrow \mathsf{Samp}(1^\lambda, p, n)$, $\boldsymbol{s} \leftarrow\!\!\$\ \mathbb{Z}_p^n$ and $\boldsymbol{r} \leftarrow\!\!\$\ \mathbb{Z}_p^n$. For an adversary $\mathcal{A}_{\mathsf{then}}$, we define its advantage in the then-challenge by

$$\begin{aligned}\mathrm{Adv}^{\mathcal{A}_{\mathsf{then}}}_{\mathsf{then}} = |&\Pr[\mathcal{A}_{\mathsf{then}}(c, h_0, h_1, h_2, G_0, G_1, h_0(\boldsymbol{s}), \mathsf{g}_1^{h_1(\boldsymbol{s})}, \mathsf{g}_2^{h_2(\boldsymbol{s})}, \mathsf{g}_2^{\boldsymbol{s}}, \mathsf{g}_1^{\boldsymbol{x}}, G_1(\boldsymbol{x}), \mathsf{g}_2^{c(\boldsymbol{s})}) = 1] \\ &- \Pr[\mathcal{A}_{\mathsf{then}}(c, h_0, h_1, h_2, G_0, G_1, h_0(\boldsymbol{s}), \mathsf{g}_1^{h_1(\boldsymbol{s})}, \mathsf{g}_2^{h_2(\boldsymbol{s})}, \mathsf{g}_2^{\boldsymbol{s}}, \mathsf{g}_1^{\boldsymbol{x}}, G_1(\boldsymbol{x}), \mathsf{g}_2^{\boldsymbol{r}}) = 1]|\end{aligned}$$

where $\boldsymbol{s} = G_0(\boldsymbol{x})$, and the probabilities are taken over $(c, h_0, h_1, h_2) \leftarrow \mathsf{Samp}(1^\lambda, p, n)$, $\boldsymbol{x} \leftarrow\!\!\$\ \mathbb{Z}_p^n$, $G_0, G_1 \leftarrow\!\!\$\ Q_{p,n}$ and $\boldsymbol{r} \leftarrow\!\!\$\ \mathbb{Z}_p^n$.

Finally, fix non-negative functions $t = t(\lambda)$, $q = q(\lambda)$, $\mu = \mu(\lambda)$. The *Recursive Decisional Diffie-Hellman* assumption (t, q, μ)-rDDH states that for all samplers of the above form that run in time t and for all then-adversaries $\mathcal{A}_{\mathsf{then}}$, there exists an if-adversary $\mathcal{A}_{\mathsf{if}}$ s.t.

$$\mathrm{Adv}_{\mathsf{if}}^{\mathcal{A}_{\mathsf{if}}}(\lambda) \geq \mathrm{Adv}_{\mathsf{then}}^{\mathcal{A}_{\mathsf{then}}}(\lambda) - \mu(\lambda) \qquad \text{and} \qquad \mathrm{time}(\mathcal{A}_{\mathsf{if}}) \leq \mathrm{time}(\mathcal{A}_{\mathsf{then}}) + q.$$

Note that we state the recursive DDH assumption in an additive manner, i.e., the advantage and time complexity of the postulated if-adversary $\mathcal{A}_{\mathsf{if}}$ only deteriorate by additive amounts. This makes the assumption very strong. We justify this by proving in Thm. 5 that any *generic* adversary on the then-challenge implies a *generic* adversary on the if-challenge with only small additive overhead.

4.2 Security under the Recursive Decisional Diffie-Hellman

Theorem 4. *Let $t = t(\lambda), q = q(\lambda), \mu = \mu(\lambda), \mu' = \mu'(\lambda)$. Under the $(t + \ell q + \ell n t_{\mathsf{pair}}, \mu)$-Decisional Square Diffie-Hellman for the target group $\mathbb{G}_2$ and (t', q, μ')-Recursive Decisional Diffie-Hellman assumptions $\mathsf{CVRF}\colon \{0,1\}^\ell \to \mathbb{G}_2^n$ is $(t, \mu + \mu'\ell)$-selectively-secure, where $t' = \Theta(\ell n^3 \log_2 p)$ and t_{pair} is the time complexity to evaluate $e : \mathbb{G}_1^2 \to \mathbb{G}_2$ once.*

Proof of Thm. 4. We will consider the *Selective Constrained Pseudorandomness* game (cf. Def. 5) for different input lengths $\mathsf{idx} \leq \ell$. By $\mathcal{A}_{\mathsf{idx}}$ we denote the adversary playing the game with inputs of length $0 \leq \mathsf{idx} \leq \ell$; its advantage is denoted by $\mathrm{Adv}_{\mathsf{CVRF\text{-}idx}}^{\mathcal{A}_{\mathsf{idx}}}$. Under the (t', q, μ')-rDDH assumption, the following claim holds:

Claim 1. *For all $1 \leq \mathsf{idx} \leq \ell$, the existence of an adversary $\mathcal{A}_{\mathsf{idx}}$ for CVRF with input length idx which runs in time at most $t + (\ell - \mathsf{idx})(q + n t_{\mathsf{pair}})$ and has advantage $\mathrm{Adv}_{\mathsf{CVRF\text{-}idx}}^{\mathcal{A}_{\mathsf{idx}}}$, implies the existence of an adversary $\mathcal{A}_{\mathsf{idx}-1}$ for CVRF with input length $\mathsf{idx} - 1$ that runs in time at most $t + (\ell - \mathsf{idx} + 1)(q + n t_{\mathsf{pair}})$ and has advantage $\mathrm{Adv}_{\mathsf{CVRF\text{-}idx}}^{\mathcal{A}_{\mathsf{idx}}} \leq \mathrm{Adv}_{\mathsf{CVRF\text{-}(idx\,-\,1)}}^{\mathcal{A}_{\mathsf{idx}-1}} + \mu'$ provided that the (t', q, μ')-rDDH assumption holds.*

We defer the claim's proof for now and prove the theorem's statement first. Consider the adversary $\mathcal{A}_\ell$ that we are interested in. Suppose that it runs in time at most t. After applying Clm. 1 to $\mathcal{A}_\ell$ for a total of ℓ times, we arrive at $\mathcal{A}_0$.

Observe that $\mathcal{A}_0$ is an adversary against the *Selective Constrained Pseudorandomness* game (cf. Def. 5) for input length 0. Since the input space only contains the empty word ϵ, $\mathcal{A}_0$ must challenge $x^* = \epsilon$. As a consequence, it must distinguish whether a value $\boldsymbol{y}^*$ is equal to $\mathsf{g}_2^{\mathsf{sk}_\epsilon^2}$ (i.e., an output of CVRF) or uniformly random while only receiving $\mathsf{vk} = \mathsf{g}_2^{\mathsf{sk}_\epsilon}$ from the game. I.e., $\mathcal{A}_0$ must solve the DSDH problem.

Moreover, $\mathcal{A}_0$ runs in time at most $t + \ell(q + n t_{\mathsf{pair}})$ since we applied Clm. 1 ℓ times. This together with the $(t + \ell(q + n t_{\mathsf{pair}}), \mu)$-$\mathsf{DSDH}$ assumption implies that $\mathrm{Adv}_{\mathsf{CVRF\text{-}0}}^{\mathcal{A}_0} \leq \mu$. As a consequence, (again, recalling the ℓ applications of Clm. 1), $\mathrm{Adv}_{\mathsf{CVRF\text{-}}\ell}^{\mathcal{A}_\ell} \leq \mu + \mu'\ell$ holds completing the proof. □

Proof of Clm. 1. For $\mathsf{idx} \in \{1, \ldots, \ell\}$, let $\mathcal{A}_{\mathsf{idx}}$ be an adversary for CVRF with inputs of length idx. In the following, we suppose that $\mathcal{A}_{\mathsf{idx}}$ runs in time at most $t + (\ell - \mathsf{idx})(q + n t_{\mathsf{pair}})$.

Further, we assume that $\mathcal{A}_{\mathsf{idx}}$ always challenges $x^* = 0 \cdots 0 \in \{0,1\}^{\mathsf{idx}}$. This is without loss of generality since the view of $\mathcal{A}_{\mathsf{idx}}$ is statistically identical (up to reordering terms) for each input $x^* \in \{0,1\}^{\mathsf{idx}}$. Indeed, we sample a fresh and independent PRG $(G_0^{(i)}, G_1^{(i)}) \leftarrow\!\!\$\ Q^2$ for every layer of the tree, so if $\mathcal{A}_{\mathsf{idx}}$ were to submit a challenge point $x^* \neq 0^{\mathsf{idx}}$, we could simply swap $G_0^{(i)}$ and $G_1^{(i)}$ whenever $x_i^* = 1$. Further, we assume—again without loss of generality—that $\mathcal{A}_{\mathsf{idx}}$ requests, besides the verification key vk, the constrained keys for the prefixes $1, 01, \ldots, 0^{\mathsf{idx}-2}1$ and the evaluation of CVRF at $0^{\mathsf{idx}-1}1$ (all including corresponding proofs). Indeed, from this information, $\mathcal{A}$ can evaluate CVRF anywhere except 0^{idx} on its own. Further, since CVRF is constraint-hiding and generates proofs and keys deterministically, $\mathcal{A}$ does not learn anything by repeatedly requesting the same evaluation or constrained key.

Hence, $\mathcal{A}_{\mathsf{idx}}$ receives the following information from its challenger: First, $\mathsf{pp}, \mathsf{vk}, \pi_\epsilon$ as sampled by $(\mathsf{pp}, \mathsf{sk}_\epsilon, \mathsf{vk}, \pi_\epsilon) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$. Here, $\mathsf{pp} = \mathcal{G} \leftarrow\!\!\$\ Q_{p,n}^{\mathsf{idx}\times 2}$, $\mathsf{sk} = \mathsf{sk}_\epsilon \leftarrow\!\!\$\ \mathbb{Z}_p^n$, $\mathsf{vk} = \mathsf{g}_2^{\mathsf{sk}_\epsilon}$, and $\pi_\epsilon = \emptyset$ where an element $\mathcal{G} \in Q_{p,n}^{\mathsf{idx}\times 2}$ is of the form

$$\mathcal{G} = ((G_0^{(1)}, G_1^{(1)}), \ldots, (G_0^{(\mathsf{idx})}, G_1^{(\mathsf{idx})}))$$

with each $G_b^{(j)} : \mathbb{Z}_p^n \to \mathbb{Z}_p^n$ being a degree 2-map. Remember that we set

$$G_\epsilon(\boldsymbol{s}) := \boldsymbol{s} \quad \text{and} \quad G_x(\boldsymbol{s}) := G_{x_{\mathsf{idx}'}}^{(\mathsf{idx}')}(\cdots(G_{x_1}^{(1)}(\boldsymbol{s})))$$

for a bit string $x = x_1 \cdots x_{\mathsf{idx}'} \in \{0,1\}^{\leq \mathsf{idx}}$. Second, it also receives the outputs of its Eval- and $\mathsf{Constrain}$-queries, given by

$$(\boldsymbol{y}, \pi) = \mathsf{Eval}_{\mathsf{pp}}(\mathsf{sk}_\epsilon, \pi_\epsilon, 0^{\mathsf{idx}-1}1) \quad \text{and} \quad (\mathsf{sk}_i, \pi_i) = \mathsf{Constrain}_{\mathsf{pp}}(\mathsf{sk}_\epsilon, \pi_\epsilon, 0^{i-1}1) \tag{1}$$

for $i \in [\mathsf{idx} - 1]$. Note that the outputs of Eval are of shape $\boldsymbol{y} = \mathsf{g}_2^{\hat{\boldsymbol{y}}^2}$ and $\pi = (\hat{\pi}, \mathsf{g}_1^{\hat{\boldsymbol{y}}})$ where $\hat{\pi} = (\mathsf{g}_1^{G_{0^i}(\mathsf{sk}_\epsilon)})_{i \in [\mathsf{idx}-1]}$ and $\hat{\boldsymbol{y}} = G_{0^{\mathsf{idx}-1}1}(\mathsf{sk}_\epsilon)$. The constrained keys are of the form $\mathsf{sk}_i = G_{0^{i-1}1}(\mathsf{sk}_\epsilon)$ and the proofs are $\pi_i = (\mathsf{g}_1^{G_{0^j}(\mathsf{sk}_\epsilon)})_{j \in [i-1]}$.

We invoke the Recursive DDH (cf. Ass. 1) assumption to show that the existence of $\mathcal{A}_{\mathsf{idx}}$ implies the existence of $\mathcal{A}_{\mathsf{idx}-1}$ as defined in the claim's statement. Concretely, we define a sampler $\mathsf{Samp}_{\mathsf{idx}}$ in Fig. 3 such that $\mathcal{A}_{\mathsf{then}} = \mathcal{A}_{\mathsf{idx}}$ which guarantees the existence of $\mathcal{A}_{\mathsf{if}} = \mathcal{A}_{\mathsf{idx}-1}$ by the assumption.
Recall that $\mathcal{A}_{\mathsf{if}}$ has to distinguish between

$$(c, h_0, h_1, h_2, G_0, G_1, h_0(\boldsymbol{s}), \mathsf{g}_1^{h_1(\boldsymbol{s})}, \mathsf{g}_2^{h_2(\boldsymbol{s})}, \mathsf{g}_2^{\boldsymbol{s}}, \mathsf{g}_2^{c(\boldsymbol{s})})$$

and

$$(c, h_0, h_1, h_2, G_0, G_1, h_0(\boldsymbol{s}), \mathsf{g}_1^{h_1(\boldsymbol{s})}, \mathsf{g}_2^{h_2(\boldsymbol{s})}, \mathsf{g}_2^{\boldsymbol{s}}, \mathsf{g}_2^{\boldsymbol{r}})$$

for $\boldsymbol{s} \leftarrow\!\!\$\ \mathbb{Z}_p^n$ and $\boldsymbol{r} \leftarrow\!\!\$\ \mathbb{Z}_p^n$. The crucial observation is that $\mathcal{A}_{\mathsf{if}}$ with $\boldsymbol{s} = \mathsf{sk}_\epsilon$ is an adversary against the CVRF construction with input length $\mathsf{idx} - 1$ (up to rearrangement of inputs): The description of (c, h_0, h_1, h_2) is $(G_0^{(i)}, G_1^{(i)})_{i \in [\mathsf{idx}-1]}$ which is equal to the public parameters $\mathsf{pp}_{\mathsf{idx}-1}$ for $\mathcal{A}_{\mathsf{idx}-1}$. Furthermore, $h_{0,\mathsf{idx}-1}(\boldsymbol{s})$, $\mathsf{g}_1^{h_{1,\mathsf{idx}-1}(\boldsymbol{s})}$ and $\mathsf{g}_2^{h_{2,\mathsf{idx}-1}(\boldsymbol{s})}$ represent the values that $\mathcal{A}_{\mathsf{idx}-1}$ receives from its Eval and $\mathsf{Constrain}$ queries, analogously to the information in Eq. (1) sent to $\mathcal{A}_{\mathsf{idx}}$. Second, $c_{\mathsf{idx}-1}(\boldsymbol{s})$ is precisely the output of

$\mathsf{Samp}_{\mathsf{idx}}(1^\lambda, p, n)$

01 Sample $(G_0^{(i)}, G_1^{(i)})_{i\in[\mathsf{idx}-1]} \leftarrow_\$ Q_{p,n}^{(\mathsf{idx}-1)\times 2}$.

02 Output $\{G_0^{(j)}, G_1^{(j)}\}_{j=1}^{\mathsf{idx}-1}$. This acts as a description of the following functions:

- $c_{\mathsf{idx}-1}\colon \mathbb{Z}_p^n \to \mathbb{Z}_p^n$ defined as $c_{\mathsf{idx}-1}(\cdot) = G_{0^{\mathsf{idx}-1}}(\cdot)^2$.
- $h_{0,\mathsf{idx}-1}\colon \mathbb{Z}_p^n \to \mathbb{Z}_p^{n\cdot(\mathsf{idx}-2)}$ defined as $h_{0,\mathsf{idx}-1}(\cdot) = (G_{0^j1}(\cdot))_{j=0}^{\mathsf{idx}-3}$, which can be computed using the degree-2 maps $\{G_0^{(j)}, G_1^{(j)}\}_{j=1}^{\mathsf{idx}}$.
- $h_{1,\mathsf{idx}-1}\colon \mathbb{Z}_p^n \to \mathbb{Z}_p^{n\cdot\mathsf{idx}}$ defined as $h_{1,\mathsf{idx}-1}(\cdot) = ((G_{0^j}(\cdot))_{j=0}^{\mathsf{idx}-2}, G_{0^{\mathsf{idx}-2}1}(\cdot))$.
- $h_{2,\mathsf{idx}-1}\colon \mathbb{Z}_p^n \to \mathbb{Z}_p^n$ defined as $h_{2,\mathsf{idx}-1}(\cdot) = G_{0^{\mathsf{idx}-2}1}(\cdot)^2$.

Figure 3: Pseudocode of $\mathsf{Samp}_{\mathsf{idx}}$.

CVRF on challenge point $x^* = 0^{\mathsf{idx}-1}$. Finally, $\mathsf{g}_1^{\boldsymbol{s}}$ in the if-challenge equals the verification key vk, which $\mathcal{A}_{\mathsf{idx}-1}$ receives.

Let us now consider $\mathcal{A}_{\mathsf{then}}$. It has to distinguish between

$$(c, h_0, h_1, h_2, G_0, G_1, h_0(\boldsymbol{s}), \mathsf{g}_1^{h_1(\boldsymbol{s})}, \mathsf{g}_2^{h_2(\boldsymbol{s})}, \mathsf{g}_2^{\boldsymbol{s}}, \mathsf{g}_1^{\boldsymbol{x}}, G_1(\boldsymbol{x}), \mathsf{g}_2^{c(\boldsymbol{s})})$$

and

$$(c, h_0, h_1, h_2, G_0, G_1, h_0(\boldsymbol{s}), \mathsf{g}_1^{h_1(\boldsymbol{s})}, \mathsf{g}_2^{h_2(\boldsymbol{s})}, \mathsf{g}_2^{\boldsymbol{s}}, \mathsf{g}_1^{\boldsymbol{x}}, G_1(\boldsymbol{x}), \mathsf{g}_2^{\boldsymbol{r}})$$

when $\boldsymbol{x} \leftarrow_\$ \mathbb{Z}_p^n$, $(G_0, G_1) \leftarrow_\$ Q_{p,n}$, $\boldsymbol{r} \leftarrow_\$ \mathbb{Z}_p^n$ and $\boldsymbol{s} = G_0(\boldsymbol{x})$. Now, $\mathcal{A}_{\mathsf{then}}$ is *almost* equal to $\mathcal{A}_{\mathsf{idx}}$, only incurring an additive overhead of $O(n)$. For this, note that G_0, G_1 play the role of $G_0^{(0)}, G_1^{(0)}$ in the construction, while the maps $G_0^{(i)}, G_1^{(i)}$ output by the sampler $\mathsf{Samp}_{\mathsf{idx}}(1^\lambda, p, n)$ play the role of $G_0^{(i+1)}, G_1^{(i+1)}$ in the construction. As a consequence, the verification key needs to be $\mathsf{g}_2^{\boldsymbol{x}}$ which can be computed from $\mathsf{g}_1^{\boldsymbol{x}}$ by evaluating n pairings.

To summarize, (t', q, μ')-rDDH implies that, if $\mathcal{A}_{\mathsf{idx}}$ exists, then there also exists $\mathcal{A}_{\mathsf{idx}-1}$ with

$$\mathcal{A}_{\mathsf{idx}-1}(\lambda) \geq \mathcal{A}_{\mathsf{idx}}(\lambda) - \mu' \qquad \text{and} \qquad \mathrm{time}(\mathcal{A}_{\mathsf{idx}-1}) \leq \mathrm{time}(\mathcal{A}_{\mathsf{idx}}) + q + nt_{\mathsf{pair}},$$

which completes the proof. □

5 Proving Recursive DDH in the Generic Group Model

Notation. For $d \in \mathbb{N}$, denote by $\mathbb{Z}_p[X]^{\leq d}$ the space of all polynomials $h(X_1, \ldots, X_n)$ of degree $\leq d$. Further, denote by $\mathbb{Z}_p[X]^d$ the space of all homogeneous polynomials of degree exactly d.

Further, for maps $G_1, \ldots, G_\ell \in Q$, denote by

$$\mathrm{span}_{\mathbb{Z}_p}[G_1, \ldots, G_\ell] \subseteq \mathbb{Z}_p[X]^{\leq 2}$$

the space generated by all polynomials that compute the outputs of $G_1, \ldots, G_\ell$.

In this section, we prove the security of the recursive DDH Assumption Ass. 1 in Maurer's generic group model assuming the hardness of the following two assumptions:

Assumption 2 (MQ)**.** The **(decisional) multivariate quadratic** assumption (t, μ)-$\mathsf{MQ}_{p,n}$ states that we have for all adversaries $\mathcal{A}$ running in time t

$$|\Pr[\mathcal{A}((G_0, G_1), (G_0(\boldsymbol{x}), G_1(\boldsymbol{x}))) = 1] - \Pr[\mathcal{A}((G_0, G_1), \boldsymbol{y}) = 1]| \leq \mu$$

where we draw $G_0, G_1 \leftarrow\!\!\$\ Q$, $\boldsymbol{x} \leftarrow\!\!\$\ \mathbb{Z}_p^n$ and $\boldsymbol{y} \leftarrow\!\!\$\ \mathbb{Z}_p^{2n}$.

Assumption 3 (MQ+)**.** Let p be prime. The **multivariate quadratic plus** assumption (t, μ)-$\mathsf{MQ}+_{p,n}$ states that we have for all adversaries $\mathcal{A}$ running in time t

$$\Pr\begin{bmatrix} h \in \mathbb{Z}_p[X_1, \ldots, X_n]^{\leq 2}, h(\boldsymbol{x}) = 0, \\ h \notin \mathrm{span}_{\mathbb{Z}_p}[G_0(X) - G_0(\boldsymbol{x}), G_1(X) - G_1(\boldsymbol{x})] \end{bmatrix} - \frac{2}{p} \leq \mu$$

where we draw $G_0, G_1 \leftarrow\!\!\$\ Q$, $\boldsymbol{x} \leftarrow\!\!\$\ \mathbb{Z}_p^n$ and $h \leftarrow \mathcal{A}((G_0, G_1), (G_0(\boldsymbol{x}), G_1(\boldsymbol{x})))$.

Note that we require the adversary's success probability in $\mathsf{MQ}+$ to be by a noticeable amount μ to be larger than $2/p$. The reason is that there exists a default adversary that has a success probability of exactly $2/p$ by simply outputting a random polynomial outside $\mathrm{span}_{\mathbb{Z}_p}[G_0(X) - G_0(\boldsymbol{x}), G_1(X) - G_1(\boldsymbol{x})]$. For example, if we always output $h(X) = (X_1 - z_1)(X_2 - z_2)$ for some $z_1, z_2 \in \mathbb{Z}_p$, then h will vanish on $\boldsymbol{x} \leftarrow\!\!\$\ \mathbb{Z}_p^n$ with probability $2/p$. On the other hand, for any *fixed* polynomial h, the probability that h vanishes on $\boldsymbol{x} \leftarrow\!\!\$\ \mathbb{Z}_p^n$ is bounded by $2/p$. For exponential p, the success probability of such adversaries is negligible, however, for polynomial p, its success probability is noticable. Hence, we need to set the advantage of an $\mathsf{MQ}+$-solver to be its success probability minus the probability of *guessing randomly correct*.

Additionally, we require the modulus p to be prime. This requirement is necessary as $\mathsf{MQ}+$ can be solved with success probability larger than $2/p$ if p is composite. For example, if p is even, then the polynomial $h(X) = \frac{p}{2} \cdot X_1$ vanishes on $\boldsymbol{x} \leftarrow\!\!\$\ \mathbb{Z}_p$ with probability at least $1/2$. We give more details on the cryptanalysis of $\mathsf{MQ}+$ in App. A.3.

Theorem 5. *Let* Samp *be a sampling algorithm with time complexity* t'' *for Ass. 1 that outputs functions* h_0, h_1, h_2, c *that can be evaluated in time* t''*. Let* $\mathcal{A}_{\mathsf{then}}$ *be a generic adversary against the* then*-challenge of Ass. 1 with advantage* $\mu + q\mu_{\mathsf{MQ}+} + \mu_{\mathsf{MQ}} + \frac{2n+3q}{p}$ *and time complexity* t *that makes* q *group-queries (in the sense of Def. 2). Assume* (t', μ_{MQ})-$\mathsf{MQ}_{p,n}$ *and* $(t', \mu_{\mathsf{MQ}+})$-$\mathsf{MQ}+_{p,n}$ *for* $t' = t + t'' + O(qn^6)$*.*

Then, there exists an adversary $\mathcal{A}_{\mathsf{if}}$ *on the* if*-challenge of Ass. 1 that makes* q *group-queries and has an advantage of* μ *and a time complexity of* t'*.*

To prove Thm. 5, we will argue by hybrid games given in Fig. 4. Each game is defined by two algorithms $(\mathsf{G}_i^{\mathcal{A}_{\mathsf{then}}}, \mathsf{R})$ that share the same state. $\mathsf{G}_i^{\mathcal{A}_{\mathsf{then}}}$ sets up the experiment and runs $\mathcal{A}_{\mathsf{then}}$. Since $\mathcal{A}_{\mathsf{then}}$ is generic, it does not receive real group elements, instead it receives handles that we represent by dummy group elements with formal variables as exponents: $\mathsf{g}_1^{H^{(1)}} = (\mathsf{g}_1^{H_1^{(1)}}, \ldots, \mathsf{g}_1^{H_{m_1}^{(1)}})$ represents the group element vector $\mathsf{g}_1^{h_1(\boldsymbol{s})} \in \mathbb{G}_1^{m_1}$, $\mathsf{g}_2^{H^{(2)}} = (\mathsf{g}_1^{H_1^{(1)}}, \ldots, \mathsf{g}_1^{H_{m_2}^{(1)}})$ represents $\mathsf{g}_2^{h_2(\boldsymbol{s})} \in \mathbb{G}_2^{m_2}$, $\mathsf{g}_2^S = (\mathsf{g}_2^{S_1}, \ldots, \mathsf{g}_2^{S_n})$ represents $\mathsf{g}_2^{\boldsymbol{s}} \in \mathbb{G}_2^n$, $\mathsf{g}_1^X = (\mathsf{g}_1^{X_1}, \ldots, \mathsf{g}_1^{X_n})$ represents $\mathsf{g}_1^{\boldsymbol{x}} \in \mathbb{G}_1^n$ and $\mathsf{g}_2^C = (\mathsf{g}_2^{C_1}, \ldots, \mathsf{g}_2^{C_n})$ reprsents $\mathsf{g}_2^{\boldsymbol{c}}$. Since $\mathcal{A}_{\mathsf{then}}$

is generic (in the sense of 2), it can only interact with the group element handles by making explicit group queries. For this end, it has to call the oracle R on a quadratic function $f \in \mathbb{Z}_p[H^{(1)}, H^{(2)}, S, X, C]$. To ensure that the adversary only multiplies the exponents of source group elements with each other, R rejects any f that is not *admissible* where the space of admissible polynomials is defined as

$$\begin{aligned} A :=& \operatorname{span}_{\mathbb{Z}_p}[V_1 \cdot V_2 \mid V_1, V_2 \in \{1, H_1^{(1)}, \ldots, H_{m_1}^{(1)}, X_1, \ldots, X_n\}] \\ &+ \operatorname{span}_{\mathbb{Z}_p}[H_1^{(2)}, \ldots, H_{m_2}^{(2)}, S_1, \ldots, S_n, C_1, \ldots, C_n]. \end{aligned}$$

Additionally, denote by $B \subseteq A$ the subspace of all admissible polynomials in which the monomials $(X_j \cdot X_k)_{j,k \in [n]}$, do not appear

$$\begin{aligned} B :=& \operatorname{span}_{\mathbb{Z}_p}[1, X_1, \ldots, X_n] + \operatorname{span}_{\mathbb{Z}_p}[X_i \cdot H_j^{(1)} \mid i \in [n], j \in [m_1]] \\ &+ \operatorname{span}_{\mathbb{Z}_p}[H_i^{(1)} \cdot H_j^{(1)} \mid i, j \in [m_1]] + \operatorname{span}_{\mathbb{Z}_p}[H_1^{(2)}, \ldots, H_{m_2}^{(2)}, S_1, \ldots, S_n, C_1, \ldots, C_n]. \end{aligned}$$

Lemma 1. *For $i \in [q]$, the advantage of $\mathcal{A}_{\mathsf{then}}$ at distinguishing G_{i-1} and G_i is bounded by $\frac{2}{p} + \mu_{\mathsf{MQ+}}$. I.e., $|\Pr[\mathsf{G}_{i-1}^{\mathcal{A}_{\mathsf{then}}}(1^\lambda) = 1] - \Pr[\mathsf{G}_i^{\mathcal{A}_{\mathsf{then}}}(1^\lambda) = 1]| \leq \frac{2}{p} + \mu_{\mathsf{MQ+}}$.*

Proof. Let $f_1, \ldots, f_q$ be the group queries made by $\mathcal{A}_{\mathsf{then}}$. G_{i-1} and G_i only differ in how they handle the i-th query f_i. We can distinguish three cases:

1. If $f_i(h_1(\boldsymbol{s}), h_2(\boldsymbol{s}), \boldsymbol{x}, \boldsymbol{s}, \boldsymbol{c}) \neq 0$, then $\mathsf{R}(f_i)$ will reply with 0 in both games.
2. If $f_i(h_1(\boldsymbol{s}), h_2(\boldsymbol{s}), X, \boldsymbol{s}, \boldsymbol{c}) \in \operatorname{span}_{\mathbb{Z}_p}[G_0(X) - \boldsymbol{s}, G_1(X) - \boldsymbol{w}]$, then $\mathsf{R}(f_i)$ will reply with 1 in both games.
3. If $f_i(h_1(\boldsymbol{s}), h_2(\boldsymbol{s}), \boldsymbol{x}, \boldsymbol{s}, \boldsymbol{c}) = 0$ and $f_i(h_1(\boldsymbol{s}), h_2(\boldsymbol{s}), X, \boldsymbol{s}, \boldsymbol{c}) \notin \operatorname{span}_{\mathbb{Z}_p}[G_0(X) - \boldsymbol{s}, G_1(X) - \boldsymbol{w}]$, then $\mathsf{R}(f_i)$ will reply with 1 in G_{i-1}, but with 0 in G_i.

It follows that $\mathcal{A}_{\mathsf{then}}$'s advantage at distinguishing G_{i-1} and G_i is bounded by the probability of the last case occurring. Let us call this probability α.

To reduce the $\mathsf{MQ+}$-problem to the problem of distinguishing G_{i-1} and G_i, we consider the following reduction: $\mathcal{R}$ receives $(G_0, G_1, G_0(\boldsymbol{x}), G_1(\boldsymbol{x}))$ from an $\mathsf{MQ+}$-challenger Ch. It sets $\boldsymbol{s} = G_0(\boldsymbol{x})$, $\boldsymbol{w} = G_1(\boldsymbol{x})$ and simulates G_{i-1} for $\mathcal{A}_{\mathsf{then}}$ by sampling $(c, h_0, h_1, h_2) \leftarrow \mathsf{Samp}(1^\lambda)$, $b \leftarrow \{0, 1\}$, and computing $c(\boldsymbol{s}), h_0(\boldsymbol{s}), h_1(\boldsymbol{s}), h_2(\boldsymbol{s})$. Note that $\mathcal{R}$ can evaluate the first $i-1$ calls of the R-oracle, since $\mathcal{R}$ is able to decide if

$$f_j(h_1(\boldsymbol{s}), h_2(\boldsymbol{s}), X, \boldsymbol{s}, \boldsymbol{c}) \in \operatorname{span}_{\mathbb{Z}_p}[G_0(X) - \boldsymbol{s}, G_1(X) - \boldsymbol{w}]$$

for $j < i$. Once $\mathcal{A}_{\mathsf{then}}$ calls R for the i-th time with f_i, $\mathcal{R}$ stops the simulation and sends

$$f'(X) := f_i(h_1(\boldsymbol{s}), h_2(\boldsymbol{s}), X, \boldsymbol{s}, \boldsymbol{c})$$

to $\mathsf{Ch}_{\mathsf{MQ+}}$. With probability α, f' is a correct $\mathsf{MQ+}$-solution, as we have

$$f'(\boldsymbol{x}) = 0, \qquad \text{but} \qquad f'(X) \notin \operatorname{span}_{\mathbb{Z}_p}[G_0(X) - G_0(\boldsymbol{x}), G_1(X) - G_1(\boldsymbol{x})].$$

Hence, we have $|\Pr[\mathsf{G}_{i-1}^{\mathcal{A}_{\mathsf{then}}}(1^\lambda) = 1] - \Pr[\mathsf{G}_i^{\mathcal{A}_{\mathsf{then}}}(1^\lambda) = 1]| \leq \alpha \leq \mu_{\mathsf{MQ+}} + \frac{2}{p}$. □

Game $\mathsf{G}_i^{\mathcal{A}_{\mathsf{then}}}(1^\lambda)$, $i \in \{0, \ldots, q+2\}$, **with adversary** $\mathcal{A}_{\mathsf{then}}$**:**

01 Initialize generic groups $\mathbb{G}_1 = \langle \mathsf{g}_1 \rangle$, $\mathbb{G}_2 = \langle \mathsf{g}_2 \rangle$
02 Set $j = 0$
03 Sample $b \leftarrow\!\!\$\ \{0,1\}$
04 Sample $(c, h_0, h_1, h_2) \leftarrow \mathsf{Samp}(1^\lambda, p, n)$
05 Sample $\boldsymbol{x} \leftarrow\!\!\$\ \mathbb{Z}_p^n$, $G_0, G_1 \leftarrow\!\!\$\ Q_{p,n}$
06 If $i \leq q$, set $\boldsymbol{s} = G_0(\boldsymbol{x})$ and $\boldsymbol{w} = G_1(\boldsymbol{x})$
07 If $i \geq q+1$, sample $\boldsymbol{s}, \boldsymbol{w} \leftarrow\!\!\$\ \mathbb{Z}_p^n$
08 If $b = 1$, sample $\boldsymbol{c} \leftarrow\!\!\$\ \mathbb{Z}_p^n$; else, set $\boldsymbol{c} = c(\boldsymbol{s})$
09 Run $b' \leftarrow \mathcal{A}_{\mathsf{then}}^{\mathsf{R}}(c, h_0, h_1, h_2, G_0, G_1, h_0(\boldsymbol{s}), \mathsf{g}_1^{H^{(1)}}, \mathsf{g}_2^{H^{(2)}}, \mathsf{g}_2^S, \mathsf{g}_1^X, \boldsymbol{w}, \mathsf{g}_2^C)$
10 Return 1, if $b = b'$; else, return 0

Group Query Subroutine $\mathsf{R}(f \in \mathbb{Z}_p[H^{(1)}, H^{(2)}, X, S, C])$**:**

01 Increase $j = j + 1$
02 If f is not admissible or $j > q$, return $\perp$
03 If $j > i \leq q+1$, do the following:
04 If $f(h_1(\boldsymbol{s}), h_2(\boldsymbol{s}), \boldsymbol{x}, \boldsymbol{s}, \boldsymbol{c}) = 0$, return 1; else, return 0;
05 If $j \leq i \leq q+1$, do the following:
06 If $f(h_1(\boldsymbol{s}), h_2(\boldsymbol{s}), X, \boldsymbol{s}, \boldsymbol{c}) \in \mathrm{span}_{\mathbb{Z}_p}[G_0(X) - \boldsymbol{s}, G_1(X) - \boldsymbol{w}]$, return 1
07 Else, return 0
08 If $i = q+2$, do the following:
09 Compute $f' \in B$ s.t. $f - f' \in \mathrm{span}_{\mathbb{Z}_p}[G_0(X) - S, G_1(X) - \boldsymbol{w}]$
10 If such an f' does not exist, return 0
11 Draw $\boldsymbol{x}' \leftarrow\!\!\$\ \mathbb{Z}_p^n$
12 If $f'(h_1(\boldsymbol{s}), h_2(\boldsymbol{s}), \boldsymbol{x}', \boldsymbol{s}, \boldsymbol{c}) = 0$, return 1; else, return 0

Figure 4: Hybrid Games for the proof of Thm. 5. Lines whose execution depends on the index i of the current hybrid have a gray background.

Lemma 2. *The advantage of* $\mathcal{A}_{\mathsf{then}}$ *at distinguishing* G_q *and* G_{q+1} *is bounded by* μ_{MQ}*. I.e.,* $|\Pr[\mathsf{G}_q^{\mathcal{A}_{\mathsf{then}}}(1^\lambda) = 1] - \Pr[\mathsf{G}_{q+1}^{\mathcal{A}_{\mathsf{then}}}(1^\lambda) = 1]| \leq \mu_{\mathsf{MQ}}$.

Proof. Let $\mathcal{R}$ be a reduction that receives $(G_0, G_1) \leftarrow\!\!\$\ Q$ and $(\boldsymbol{s}, \boldsymbol{w})$ from an MQ-challenger Ch. $\mathcal{R}$ simulates the games G_q and G_{q+1} for $\mathcal{A}_{\mathsf{then}}$ by computing $h_0(\boldsymbol{s}), h_1(\boldsymbol{s}), h_2(\boldsymbol{s}), c(\boldsymbol{s})$, sampling $b \leftarrow\!\!\$\ \{0,1\}$ and setting $\boldsymbol{c}$ to either $c(\boldsymbol{s})$ or sampling it uniformly at random. Note that $\mathcal{R}$ can answer each group query made by $\mathcal{A}$ to R, as it can check if $f(h_1(\boldsymbol{s}), h_2(\boldsymbol{s}), X, \boldsymbol{s}, \boldsymbol{c}) \in \mathrm{span}_{\mathbb{Z}_p}[G_0(X) - \boldsymbol{s}, G_1(X) - \boldsymbol{w}]$.

Now, if the vector $(\boldsymbol{s}, \boldsymbol{w})$ that $\mathcal{R}$ received from Ch equals $(G_0(\boldsymbol{x}), G_1(\boldsymbol{x}))$ for some $\boldsymbol{x} \leftarrow\!\!\$\ \mathbb{Z}_p^n$, then $\mathcal{R}$ simulates the game G_q for $\mathcal{A}_{\mathsf{then}}$. On the other hand, if $(\boldsymbol{s}, \boldsymbol{w}) \leftarrow\!\!\$\ \mathbb{Z}_p^n$ has been sampled uniformly at random, then $\mathcal{R}$ simulates the game G_{q+1} for $\mathcal{A}_{\mathsf{then}}$. Hence, $\mathcal{R}$ can use the difference in $\mathcal{A}_{\mathsf{then}}$'s win-probability to decide its decisional MQ-problem. Hence, we have $|\Pr[\mathsf{G}_q^{\mathcal{A}_{\mathsf{then}}}(1^\lambda) = 1] - \Pr[\mathsf{G}_{q+1}^{\mathcal{A}_{\mathsf{then}}}(1^\lambda) = 1]| \leq \mu_{\mathsf{MQ}}$. □

Lemma 3. *Let $n \geq 3$. Over the randomness of $(G_0, G_1) \leftarrow\!\!\$\ Q$, we have*

$$\Pr[\exists \boldsymbol{s}, \boldsymbol{w} \in \mathbb{Z}_p^n, \exists f \in \mathrm{span}_{\mathbb{Z}_p}[G_0(X) - \boldsymbol{s}, G_1(X) - \boldsymbol{w}] \mid f \neq 0, \deg f \leq 1] \leq \frac{2n}{p}.$$

Proof. Let $\boldsymbol{s}, \boldsymbol{w} \in \mathbb{Z}_p^n$ be arbitrary and let $g_1, \ldots, g_{2n} \in \mathbb{Z}_p[X]^{\leq 2}$ be the polynomials computing $(G_0(X) - \boldsymbol{s}, G_1(X) - \boldsymbol{w})$. Denote by $g_1^\top, \ldots, g_{2n}^\top \in \mathbb{Z}_p[X]^2$ their top terms, i.e., $g_i^\top$ is the homogeneous degree-2 part of g_i. Note that $g_1^\top, \ldots, g_{2n}^\top$ are independent of $\boldsymbol{s}$ and $\boldsymbol{w}$. The space $\mathrm{span}_{\mathbb{Z}_p}[G_0(X) - \boldsymbol{s}, G_1(X) - \boldsymbol{w}]$ can only contain a degree-1 polynomial if the top parts cancel out, i.e., if there is some $\alpha \in \mathbb{Z}_p^{2n}$, $\alpha \neq 0$, s.t.

$$\alpha_1 \cdot g_1^\top(X) + \cdots + \alpha_{2n} \cdot g_{2n}^\top(X) = 0.$$

Since G_0, G_1 are uniformly at random sampled from Q, their top terms $g_1^\top, \ldots, g_{2n}^\top$ are distributed uniformly at random in $\mathbb{Z}_p[X]^2$. The vector space dimension of $\mathbb{Z}_p[X]^2$ is given by $\dim_{\mathbb{Z}_p}(\mathbb{Z}_p[X]^2) = \binom{n+1}{2}$. Since $\binom{n+1}{2} \geq 2n$, by a Schwartz-Zippel argument, the probability that $g_1^\top, \ldots, g_{2n}^\top$ are linearly dependent is bounded by $2n/p$. □

Lemma 4. *The advantage of $\mathcal{A}_{\mathsf{then}}$ at distinguishing G_{q+1} and G_{q+2} is bounded by $(2n+q)/p$. I.e., $|\Pr[\mathsf{G}_{q+1}^{\mathcal{A}_{\mathsf{then}}}(1^\lambda) = 1] - \Pr[\mathsf{G}_{q+2}^{\mathcal{A}_{\mathsf{then}}}(1^\lambda) = 1]| \leq (2n+q)/p$.*

Proof. G_{q+1} and G_{q+2} only differ in how they handle group queries. We show that this difference is statistically negligible.

Because of Lem. 3, with probability $\geq 1 - 2n/p$, the function $(G_0, G_1) \leftarrow\!\!\$\ Q^2$ is sampled such that $\mathrm{span}_{\mathbb{Z}_p}[G_0(X) - \boldsymbol{s}, G_1(X) - \boldsymbol{w}]$ contains no polynomials that are linear in the X-variables (for all $\boldsymbol{s}, \boldsymbol{w} \in \mathbb{Z}_p^n$). In this case, we claim that the statistical distance between the outputs of R in both games are bounded by $1/p$.

Let $f \in A$. First, assume $f \notin B + \mathrm{span}_{\mathbb{Z}_p}[G_0(X) - S, G_1(X) - \boldsymbol{w}]$. In this case, R outputs 0 in G_{q+2} We claim that R in G_{q+1} must also output 0, as $f(h_1(\boldsymbol{s}), h_2(\boldsymbol{s}), X, \boldsymbol{s}, \boldsymbol{c}) \notin \mathrm{span}_{\mathbb{Z}_p}[G_0(X) - \boldsymbol{s}, G_1(X) - \boldsymbol{w}]$. Indeed, modulo B we can eliminate all monomials of f that are not of shape $X_j \cdot X_k$, $j, k \in [n]$, and deduce $f \notin \mathrm{span}_{\mathbb{Z}_p}[G_0(X), G_1(X)] + B$. Now, if $f(h_1(\boldsymbol{s}), h_2(\boldsymbol{s}), X, \boldsymbol{s}, \boldsymbol{c}) \in \mathrm{span}_{\mathbb{Z}_p}[G_0(X) - \boldsymbol{s}, G_1(X) - \boldsymbol{w}]$, we could again eliminate all monomials that are not $X_j \cdot X_k$, and extract the contradicting statement $f \in \mathrm{span}_{\mathbb{Z}_p}[G_0(X), G_1(X)] + B$. Hence, R must also output 0 in G_{q+1}.

Now, let $f' \in B$ s.t. $f - f' \in \mathrm{span}_{\mathbb{Z}_p}[G_0(X) - S, G_1(X) - \boldsymbol{w}]$. Because of Lem. 3, $\mathrm{span}_{\mathbb{Z}_p}[G_0(X) - \boldsymbol{s}, G_1(X) - \boldsymbol{w}]$ does not contain degree-1 polynomials. Hence, the degree-2 polynomial $f(h_1(\boldsymbol{s}), h_2(\boldsymbol{s}), X, \boldsymbol{s}, \boldsymbol{c})$ can only lie in $\mathrm{span}_{\mathbb{Z}_p}[G_0(X) - \boldsymbol{s}, G_1(X) - \boldsymbol{w}]$ if the degree-1 polynomial $f'(h_1(\boldsymbol{s}), h_2(\boldsymbol{s}), X, \boldsymbol{s}, \boldsymbol{c})$ is zero. By a Schwartz-Zippel argument, we have $f'(h_1(\boldsymbol{s}), h_2(\boldsymbol{s}), X, \boldsymbol{s}, \boldsymbol{c}) = 0$ if $f'(h_1(\boldsymbol{s}), h_2(\boldsymbol{s}), \boldsymbol{x}', \boldsymbol{s}, \boldsymbol{c}) = 0$ for $\boldsymbol{x}' \leftarrow\!\!\$\ \mathbb{Z}_p^n$, except with probability $1/p$. Since the output of R in G_{q+1} and G_{q+2} depends on $f(h_1(\boldsymbol{s}), h_2(\boldsymbol{s}), X, \boldsymbol{s}, \boldsymbol{c}) \in \mathrm{span}_{\mathbb{Z}_p}[G_0(X) - \boldsymbol{s}, G_1(X) - \boldsymbol{w}]$ and $f'(h_1(\boldsymbol{s}), h_2(\boldsymbol{s}), \boldsymbol{x}', \boldsymbol{s}, \boldsymbol{c}) = 0$, respectively, the statistical distance in its output behavior is bounded by $1/p$. □

Proof of Thm. 5. Let $\mathcal{A}_{\mathsf{then}}$ be an adversary on the then-challenge of Ass. 1 with time complexity t and advantage $\mu + q\left(\mu_{\mathsf{MQ}+} + \frac{2}{p}\right) + \mu_{\mathsf{MQ}} + \frac{2n+q}{p}$. Because of Lems. 1, 2 and 4, $\mathcal{A}_{\mathsf{then}}$'s advantage at winning G_{q+2} is at least μ.

Let Ch be a challenger for the if-challenge of Ass. 1. We give a reduction $\mathcal{R}$ that gets challenged by Ch and simulates the then-game for $\mathcal{A}_{\mathsf{then}}$. Let $\mathbb{A}_1 = \langle \mathsf{a}_1 \rangle$, $\mathbb{A}_2 = \langle \mathsf{a}_2 \rangle$ be the

p-order groups used by Ch. At the start of the if-game, Ch samples $\boldsymbol{s} \leftarrow\!\!\$\ \mathbb{Z}_p^n$, $b \leftarrow\!\!\$\ \{0,1\}$ and $(h_0, h_1, h_2, c) \leftarrow \mathsf{Samp}(1^\lambda)$. If $b = 0$, it sets $\boldsymbol{c} = c(\boldsymbol{s})$, otherwise, it samples $\boldsymbol{c} \leftarrow\!\!\$\ \mathbb{Z}_p^n$. It initializes $\mathcal{R}$ by sending $c, h_0, h_1, h_2, h_0(\boldsymbol{s}), \mathsf{a}_1^{h_1(\boldsymbol{s})}, \mathsf{a}_2^{h_2(\boldsymbol{s})}, \mathsf{a}_2^{\boldsymbol{s}}, \mathsf{a}_2^{\boldsymbol{c}}$ to it.

As $\mathcal{R}$ cannot control $\mathbb{A}_1, \mathbb{A}_2$, it has to set up its own generic groups $\mathbb{G}_1 = \langle \mathsf{g}_1 \rangle$, $\mathbb{G}_2 = \langle \mathsf{g}_2 \rangle$. $\mathcal{R}$ samples $G_0, G_1 \leftarrow\!\!\$\ Q$, $\boldsymbol{w} \leftarrow\!\!\$\ \mathbb{Z}_p^n$ and sends $c, h_0, h_1, h_2, G_0, G_1, h_0(\boldsymbol{s}), \mathsf{g}_1^{H^{(1)}}, \mathsf{g}_2^{H^{(2)}}, \mathsf{g}_2^S, \mathsf{g}_1^X, \boldsymbol{w}, \mathsf{g}_2^c$ to $\mathcal{A}_{\mathsf{then}}$, while simulating the game G_{q+2} to $\mathcal{A}_{\mathsf{then}}$.

Whenever $\mathcal{A}_{\mathsf{then}}$ makes an admissible group query $f \in A$, $\mathcal{R}$ computes the output of $\mathsf{R}(f)$ as follows: $\mathcal{R}$ computes the polynomial $f' \in B$ s.t. $f - f' \in \mathrm{span}_{\mathbb{Z}_p}[G_0(X) - S, G_1(X) - \boldsymbol{w}]$ (if such an f' does not exist, $\mathcal{R}$ returns 0), samples $\boldsymbol{x}' \leftarrow\!\!\$\ \mathbb{Z}_p^n$ and checks the equality $f'(h_1(\boldsymbol{s}), h_2(\boldsymbol{s}), \boldsymbol{x}', \boldsymbol{s}, \boldsymbol{c}) = 0$ by verifying

$$\mathsf{a}_2^{f'(h_1(\boldsymbol{s}), h_2(\boldsymbol{s}), \boldsymbol{x}', \boldsymbol{s}, \boldsymbol{c})} = \mathsf{a}_2^0.$$

$\mathcal{R}$ can indeed verify the above equality of group elements as it is given the group elements $\mathsf{a}_1^{h_1(\boldsymbol{s})}, \mathsf{a}_2^{h_2(\boldsymbol{s})}, \mathsf{a}_2^{\boldsymbol{s}}, \mathsf{a}_2^{\boldsymbol{c}}$ by Ch and knows $\boldsymbol{x}'$ in plain.

Finally, note that, for each group query made by $\mathcal{A}_{\mathsf{then}}$, $\mathcal{R}$ makes exactly one group query to Ch. Hence, $\mathcal{R}$ makes at most q group queries in total. □

References

[ACF09] Michel Abdalla, Dario Catalano, and Dario Fiore. Verifiable random functions from identity-based key encapsulation. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 554–571. Springer, Berlin, Heidelberg, April 2009.

[AFI+24] Rika Akiyama, Hiroki Furue, Yasuhiko Ikematsu, Fumitaka Hoshino, Koha Kinjo, Haruhisa Kosuge, Satoshi Nakamura, Shingo Orihara, Tsuyoshi Takagi, and Kimihiro Yamakoshi. QR-UOV. Technical report, National Institute of Standards and Technology, 2024. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures.

[AFP16] Hamza Abusalah, Georg Fuchsbauer, and Krzysztof Pietrzak. Constrained PRFs for unbounded inputs. In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 413–428. Springer, Cham, February / March 2016.

[AMYY25] Shweta Agrawal, Anuja Modi, Anshu Yadav, and Shota Yamada. Evasive LWE: Attacks, variants & obfustopia. Cryptology ePrint Archive, Report 2025/375, 2025.

[AWY20] Shweta Agrawal, Daniel Wichs, and Shota Yamada. Optimal broadcast encryption from LWE and pairings in the standard model. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 149–178. Springer, Cham, November 2020.

[Bar04] Magali Bardet. *Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie.* Theses, Université Pierre et Marie Curie - Paris VI, December 2004.

[BBFR24] Ryad Benadjila, Charles Bouillaguet, Thibauld Feneuil, and Matthieu Rivain. MQOM — MQ on my Mind. Technical report, National Institute of Standards and Technology, 2024. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures.

[BCC+24] Ward Beullens, Fabio Campos, Sofía Celi, Basil Hess, and Matthias J. Kannwischer. MAYO. Technical report, National Institute of Standards and Technology, 2024. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures.

[BCD+24] Ward Beullens, Ming-Shing Chen, Jintai Ding, Boru Gong, Matthias J. Kannwischer, Jacques Patarin, Bo-Yuan Peng, Dieter Schmidt, Cheng-Jhih Shih, Chengdong Tao, and Bo-Yin Yang. UOV — Unbalanced Oil and Vinegar. Technical report, National Institute of Standards and Technology, 2024. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures.

[BCKL09] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. Compact e-cash and simulatable VRFs revisited. In Hovav Shacham and Brent Waters, editors, *PAIRING 2009*, volume 5671 of *LNCS*, pages 114–131. Springer, Berlin, Heidelberg, August 2009.

[BFS03] Magali Bardet, Jean-Charles Faugère, and Bruno Salvy. Complexity of Gröbner basis computation for Semi-regular Overdetermined sequences over F_2 with solutions in F_2. Research Report RR-5049, INRIA, 2003.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 501–519. Springer, Berlin, Heidelberg, March 2014.

[BHKÜ22] Nicholas Brandt, Dennis Hofheinz, Julia Kastner, and Akin Ünal. The price of verifiability: Lower bounds for verifiable random functions. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part II*, volume 13748 of *LNCS*, pages 747–776. Springer, Cham, November 2022.

[BHLS24] Dan Boneh, Iftach Haitner, Yehuda Lindell, and Gil Segev. Exponent-VRFs and their applications. Cryptology ePrint Archive, Paper 2024/397, 2024.

[BHLS25] Dan Boneh, Iftach Haitner, Yehuda Lindell, and Gil Segev. Exponent-vrfs and their applications. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology – EUROCRYPT 2025*, pages 195–224, Cham, 2025. Springer Nature Switzerland.

[Bit17] Nir Bitansky. Verifiable random functions from non-interactive witness-indistinguishable proofs. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part II*, volume 10678 of *LNCS*, pages 567–594. Springer, Cham, November 2017.

[Bit20] Nir Bitansky. Verifiable random functions from non-interactive witness-indistinguishable proofs. *Journal of Cryptology*, 33(2):459–493, April 2020.

[BMR10] Dan Boneh, Hart William Montgomery, and Ananth Raghunathan. Algebraic pseudorandom functions with improved efficiency from the augmented cascade. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 2010*, pages 131–140. ACM Press, October 2010.

[BÜW24] Chris Brzuska, Akin Ünal, and Ivy K. Y. Woo. Evasive LWE assumptions: Definitions, classes, and counterexamples. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part IV*, volume 15487 of *LNCS*, pages 418–449. Springer, Singapore, December 2024.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 280–300. Springer, Berlin, Heidelberg, December 2013.

[CG21] Alessio Caminata and Elisa Gorla. Solving degree, last fall degree, and related invariants. Cryptology ePrint Archive, Report 2021/1611, 2021.

[CKPS00] Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 392–407. Springer, Berlin, Heidelberg, May 2000.

[CRV14] Nishanth Chandran, Srinivasan Raghuraman, and Dhinakaran Vinayagamurthy. Constrained pseudorandom functions: Verifiable and delegatable. Cryptology ePrint Archive, Report 2014/522, 2014.

[Dat20] Pratish Datta. Constrained pseudorandom functions from functional encryption. *Theoretical Computer Science*, 809:137–170, 2020.

[DDM17] Pratish Datta, Ratna Dutta, and Sourav Mukhopadhyay. Constrained pseudorandom functions for unconstrained inputs revisited: Achieving verifiability and key delegation. In Serge Fehr, editor, *PKC 2017, Part II*, volume 10175 of *LNCS*, pages 463–493. Springer, Berlin, Heidelberg, March 2017.

[DDM19] Pratish Datta, Ratna Dutta, and Sourav Mukhopadhyay. Constrained pseudorandom functions for turing machines revisited: How to achieve verifiability and key delegation. *Algorithmica*, 81(9):3245–3390, May 2019.

[DG10] Vivien Dubois and Nicolas Gama. The degree of regularity of HFE systems. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 557–576. Springer, Berlin, Heidelberg, December 2010.

[DGKR18] Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 66–98. Springer, Cham, April / May 2018.

[DJJ24] Quang Dao, Aayush Jain, and Zhengzhong Jin. Non-interactive zero-knowledge from LPN and MQ. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part IX*, volume 14928 of *LNCS*, pages 321–360. Springer, Cham, August 2024.

[DJM+25] Nico Döttling, Abhishek Jain, Giulio Malavolta, Surya Mathialagan, and Vinod Vaikuntanathan. Simple and general counterexamples for private-coin evasive LWE. Cryptology ePrint Archive, Report 2025/374, 2025.

[DKW16] Apoorvaa Deshpande, Venkata Koppula, and Brent Waters. Constrained pseudorandom functions for unconstrained inputs. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 124–153. Springer, Berlin, Heidelberg, May 2016.

[DY05] Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In Serge Vaudenay, editor, *PKC 2005*, volume 3386 of *LNCS*, pages 416–431. Springer, Berlin, Heidelberg, January 2005.

[Fau02] Jean Charles Faugère. A new efficient algorithm for computing gröbner bases without reduction to zero (f5). In *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*, ISSAC '02, page 75–83, New York, NY, USA, 2002. Association for Computing Machinery.

[FBS04] Jean-Charles Faugère, Magali Bardet, and Bruno Salvy. On the complexity of gröbner basis computation of semi-regular overdetermined algebraic equations. In *Proceedings of the International Conference on Polynomial System Solving*, 2004.

[Frö85] Ralf Fröberg. An inequality for hilbert series of graded algebras. *MATHEMATICA SCANDINAVICA*, 56:117–144, Dec. 1985.

[Fuc14] Georg Fuchsbauer. Constrained verifiable random functions. In Michel Abdalla and Roberto De Prisco, editors, *SCN 14*, volume 8642 of *LNCS*, pages 95–114. Springer, Cham, September 2014.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986.

[GHKW17] Rishab Goyal, Susan Hohenberger, Venkata Koppula, and Brent Waters. A generic approach to constructing and proving verifiable random functions. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part II*, volume 10678 of *LNCS*, pages 537–566. Springer, Cham, November 2017.

[GHM+17] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, page 51–68, New York, NY, USA, 2017. Association for Computing Machinery.

[HHY25] Tzu-Hsiang Huang, Wei-Hsiang Hung, and Shota Yamada. A note on obfuscation-based attacks on private-coin evasive LWE. Cryptology ePrint Archive, Report 2025/421, 2025.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[HJ16] Dennis Hofheinz and Tibor Jager. Verifiable random functions from standard assumptions. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 336–362. Springer, Berlin, Heidelberg, January 2016.

[HJL25] Yao-Ching Hsieh, Aayush Jain, and Huijia Lin. Lattice-based post-quantum iO from circular security with random opening assumption (part II: zeroizing attacks against private-coin evasive LWE assumptions). Cryptology ePrint Archive, Report 2025/390, 2025.

[HKK23] Dennis Hofheinz, Julia Kastner, and Karen Klein. The power of undirected rewindings for adaptive security. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part II*, volume 14082 of *LNCS*, pages 725–758. Springer, Cham, August 2023.

[HNS25] Adam Blatchley Hansen, Jesper Buus Nielsen, and Mark Simkin. Ocash: Fully anonymous payments between blockchain light clients. In Tibor Jager and Jiaxin Pan, editors, *Public-Key Cryptography – PKC 2025*, pages 169–202, Cham, 2025. Springer Nature Switzerland.

[HW10] Susan Hohenberger and Brent Waters. Constructing verifiable random functions with large input spaces. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 656–672. Springer, Berlin, Heidelberg, May / June 2010.

[Jag15] Tibor Jager. Verifiable random functions from weaker assumptions. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 121–143. Springer, Berlin, Heidelberg, March 2015.

[Kat17] Shuichi Katsumata. On the untapped potential of encoding predicates by arithmetic circuits and their applications. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part III*, volume 10626 of *LNCS*, pages 95–125. Springer, Cham, December 2017.

[Koh19] Lisa Kohl. Hunting and gathering - verifiable random functions from standard assumptions with short proofs. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 408–437. Springer, Cham, April 2019.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 669–684. ACM Press, November 2013.

[Lev85] Leonid A. Levin. One-way functions and pseudorandom generators. In *17th ACM STOC*, pages 363–365. ACM Press, May 1985.

[LLC15] Bei Liang, Hongda Li, and Jinyong Chang. Constrained verifiable random functions from indistinguishability obfuscation. In Man Ho Au and Atsuko Miyaji, editors, *ProvSec 2015*, volume 9451 of *LNCS*, pages 43–60. Springer, Cham, November 2015.

[Lys02] Anna Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 597–612. Springer, Berlin, Heidelberg, August 2002.

[LZW19] Muhua Liu, Ping Zhang, and Qingtao Wu. A novel construction of constrained verifiable random functions. *Security and Communication Networks*, 2019(1):4187892, 2019.

[Mau05] Ueli M. Maurer. Abstract models of computation in cryptography (invited paper). In Nigel P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *LNCS*, pages 1–12. Springer, Berlin, Heidelberg, December 2005.

[MDB08] Mohamed Saied Emam Mohamed, Jintai Ding, and Johannes Buchmann. Algebraic cryptanalysis of MQQ public key cryptosystem by MutantXL. Cryptology ePrint Archive, Report 2008/451, 2008.

[MRV99] Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *40th FOCS*, pages 120–130. IEEE Computer Society Press, October 1999.

[Nie21] David Niehues. Verifiable random functions with optimal tightness. In Juan Garay, editor, *PKC 2021, Part II*, volume 12711 of *LNCS*, pages 61–91. Springer, Cham, May 2021.

[Par10] Keith Pardue. Generic sequences of polynomials. *Journal of Algebra*, 324(4):579–590, 2010.

[Ros18] Razvan Rosie. Adaptive-secure VRFs with shorter keys from static assumptions. In Jan Camenisch and Panos Papadimitratos, editors, *CANS 18*, volume 11124 of *LNCS*, pages 440–459. Springer, Cham, September / October 2018.

[Sal23] Flavio Salizzoni. An upper bound for the solving degree in terms of the degree of regularity, 2023.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Berlin, Heidelberg, May 1997.

[Tsa22] Rotem Tsabary. Candidate witness encryption from lattice techniques. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 535–559. Springer, Cham, August 2022.

[Ünal23a] Akin Ünal. New baselines for local pseudorandom number generators by field extensions. Cryptology ePrint Archive, Report 2023/550, 2023.

[Ünal23b] Akin Ünal. Worst-case subexponential attacks on PRGs of constant degree or constant locality. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part I*, volume 14004 of *LNCS*, pages 25–54. Springer, Cham, April 2023.

[VWW22] Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Witness encryption and null-IO from evasive LWE. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part I*, volume 13791 of *LNCS*, pages 195–221. Springer, Cham, December 2022.

[WCD+24] Lih-Chung Wang, Chun-Yen Chou, Jintai Ding, Yen-Liang Kuan, Jan Adriaan Leegwater, Ming-Siou Li, Bo-Shu Tseng, Po-En Tseng, and Chia-Chun Wang. SNOVA. Technical report, National Institute of Standards and Technology, 2024. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures.

[Wee22] Hoeteck Wee. Optimal broadcast encryption and CP-ABE from evasive lattice assumptions. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 217–241. Springer, Cham, May / June 2022.

[Yam17] Shota Yamada. Asymptotically compact adaptively secure lattice IBEs and verifiable random functions via generalized partitioning techniques. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 161–193. Springer, Cham, August 2017.

[ZLX23] Yao Zan, Hongda Li, and Haixia Xu. Adaptively secure constrained verifiable random function. In Moti Yung, Chao Chen, and Weizhi Meng, editors, *Science of Cyber Security*, pages 367–385, Cham, 2023. Springer Nature Switzerland.

A Cryptanalysis

We discuss here cryptanalysis on the assumptions on which we base the security of our construction in § 3.

A.1 Cryptanalysis of Recursive and Evasive Assumptions

As already pointed out, our recursive DDH assumption shares a similar structure with lattice-based evasive assumptions [VWW22, Tsa22, Wee22]. This structure basically consists of two challenges, an if- and a then-challenge, and postulating the hardness of the

then-challenge conditioned on the if-challenge being hard. Additionally, both assumptions postulate their claims for arbitrary PPT sampling algorithms Samp. Recently, evasive lattice-based [VWW22, Tsa22, Wee22] assumptions have aroused a lot of suspicion as counterexamples for evasive Learning with Errors [BÜW24, HHY25, HJL25, AMYY25, DJM+25] have been published.

We want to point out that none of the ideas underlying counterexamples against evasive Learning With Errors (LWE) are applicable to our recursive DDH assumption:

- First, note that all attacks on evasive LWE work, basically, by hiding low-rank matrices with some low-norm noise. These kinds of techniques are not applicable to our assumption, as recursive DDH does not introduce matrices or noise.
- Also, we want to point out again that we proved the soundness of recursive DDH in the generic group model under MQ and MQ+, while evasive LWE does not enjoy proofs in idealized models under contained falsifiable assumptions. A successful attack on recursive DDH would, hence, imply either an attack on MQ or MQ+, or make non-generic use of the underlying groups.

A.2 Cryptanalysis of MQ

The hardness of the MQ problem is notorious in cryptography. It underlies the security of post-quantum NIZK protocols [DJJ24] and digital signature schemes [WCD+24, BCD+24, AFI+24, BCC+24, BBFR24]. It is known that its search version can be reduced to the decisional version if the underlying modulus is polynomial [Üna23b, Üna23a]. In summary, to break the decisional MQ problem algebraically, its homogenized version needs to have a low[12] degree of regularity [BFS03, Bar04, FBS04]. To estimate the degree of regularity, one assumes that the system is semi-regular [Frö85, Par10].[13] In this case, its Hilbert-series is given by $\frac{(1-T^2)^{2n}}{(1-T)^{n+1}}$. Since the degree of regularity of this series is linear [BFS03], any algebraic attack on MQ has a time complexity in $2^{\Omega(n)}$. (For non-semi-regular sequences, the degree of regularity may be very low, or infinite.)

A.3 Cryptanalysis of MQ+

Let $\mathcal{A}$ be some adversary for the MQ+ assumption over some prime modulus p. Recall that we defined $\mathcal{A}$'s advantage as follows:

$$\mathrm{Adv}^{\mathcal{A}}_{\mathsf{MQ}+,p,n} := \Pr\begin{bmatrix} h \in \mathbb{Z}_p[X_1,\ldots,X_n]^{\leq 2}, h(\boldsymbol{x}) = 0, \\ h \notin \mathrm{span}_{\mathbb{Z}_p}[G_0(X) - G_0(\boldsymbol{x}), G_1(X) - G_1(\boldsymbol{x})] \end{bmatrix} - \frac{2}{p}\,.$$

where $G_0, G_1 \leftarrow\!\!\$\ Q, \boldsymbol{x} \leftarrow\!\!\$\ \mathbb{Z}_p^n, h \leftarrow \mathcal{A}((G_0,G_1),(G_0(\boldsymbol{x}),G_1(\boldsymbol{x})))$. We will first discuss some simple adversaries for MQ+ that basically *guess intelligently* and achieve a success

[12] Concretely, the time complexity of most algebraic algorithms for solving MQ is exponential in the degree of regularity [Sal23]. So, the degree of regularity would need to be constant for poly-time attacks, and sublinear for subexponential-time attacks.

[13] A sequence $f_1,\ldots,f_m$ of quadratic homogeneous polynomials is called *semi-regular* if the spaces $\mathrm{span}_{\mathbb{Z}_p}\left[X_{i_1}\cdots X_{i_{d-2}}\cdot f_j | i_1,\ldots,i_d \in [n], j \in [m]\right]$ admit as few linear dependencies as possible (where d goes from 2 until the degree of regularity).

probability of almost $2/p$. For this reason we subtract $2/p$ from the adversary's success probability, note that whenever the modulus is superpolynomial this term is negligible. Afterwards, we will discuss algebraic attacks on MQ+, and, finally, we will give a heuristic reduction on it, assuming that we have an LPN oracle over $\mathbb{Z}_p$.

A.3.1 On Guessing Solvers.

For *prime* modulus $p > 2$, a straightforward attack on MQ+ is given by choosing $i, j \in [n]$, $i \neq j$, s.t. the monomial $X_i \cdot X_j$ does not lie in $\mathrm{span}_{\mathbb{Z}_p}[G_0(X) - G_0(\boldsymbol{x}), G_1(X) - G_1(\boldsymbol{x})]$ and simply outputting $h(X) = X_i X_j$ as solution. Indeed, the probability h vanishing on $\boldsymbol{x} \leftarrow \mathbb{Z}_p^n$ is exactly $\frac{2}{p} - \frac{1}{p^2}$.[14] However, since we subtract the trivial amount of $2/p$ from the success probability of each adversary, this approach does not yield a positive advantage.

On the other hand, the Schwartz-Zippel lemma guarantees that the success probability of any adversary that only considers the input (G_0, G_1) and ignores $(G_0(\boldsymbol{x}), G_1(\boldsymbol{x}))$ while computing $h \in \mathbb{Z}_p[X]$ will have a success probability of at most $2/p$, amounting again to a non-positive advantage. (This is, since h is independent of $\boldsymbol{x}$ in that case, and hence we have $\Pr_{\boldsymbol{x} \leftarrow \$ \mathbb{Z}_p^n}[h(x) = 0] \leq 2/p$.) We can deduce from this that any meaningful solver really needs to take advantage from $G = (G_0, G_1)$ *and* $G(\boldsymbol{x}) = (G_0(\boldsymbol{x}), G_1(\boldsymbol{x}))$.[15]

Now, if p is a *composite* modulus, it is feasible to get an advantage significantly higher than $2/p$. For example, if $p = q \cdot q'$ is the product of two numbers q and q', an adversary can simply output $h(X) = q' \cdot X_i \cdot X_j$ as solution (where it chooses $i \neq j$ such that $q' \cdot X_i \cdot X_j$ does not lie in $\mathrm{span}_{\mathbb{Z}_p}[G(X) - G(\boldsymbol{x})]$). This polynomial will vanish on $\boldsymbol{x} \leftarrow\$ \mathbb{Z}_p^n$ with probability $\frac{2}{q} - \frac{1}{q^2}$, which is larger than $\frac{2}{p}$ by a factor of almost q'. This is the reason why we state the MQ+ assumption *only* for prime modulus p.

A.3.2 On Algebraic Solvers.

To bound the time complexity of algebraic algorithms for breaking MQ+, we assume again that $2n$ uniformly random quadratic polynomials $(g_1, \ldots, g_{2n}) = (G_0, G_1) \leftarrow\$ Q$ form a *semi-regular* sequence with overwhelming probability.[16] Now, for an algebraic algorithm to deduce a new degree-2 equation h from the system of equations $(g_1, \ldots, g_{2n})$, it would need to find at least one so-called non-trivial *syzygy* [DG10, CG21]. However, semi-regular sequences do not admit non-trivial syzygies until the degree of regularity.[17] This means, any algebraic algorithm on MQ+ will need to search until the degree of regularity, before it can find a solution for the MQ+ problem. Under semi-regularity, the degree of regularity for $g_1, \ldots, g_{2n}$ grows linearly in n, hence implying an exponential time complexity for algorithms based on Macaulay matrices and S-polynomials [CKPS00, Fau02, MDB08]

[14] Another approach could be to choose $i \in [n]$ such that $h_i(X) = X_i \cdot (X_i - 1)$ does not lie in $\mathrm{span}_{\mathbb{Z}_p}[G_0(X) - G_0(\boldsymbol{x}), G_1(X) - G_1(\boldsymbol{x})]$. In that case, h_i would vanish with probability exactly $2/p$ on $\boldsymbol{x} \leftarrow\$ \mathbb{Z}_p^n$. However, with negligible probability it may happen that all $h_1, \ldots, h_n$ are contained in $\mathrm{span}_{\mathbb{Z}_p}[G_0(X) - G_0(\boldsymbol{x}), G_1(X) - G_1(\boldsymbol{x})]$.

[15] Indeed, if we go back to the solver that outputs some $h(X) = X_i X_j$ as solution, note that this adversary does not need to take $G(\boldsymbol{x})$ into account. In fact, it suffices for it to know G to find $i, j \in [n]$ s.t. $X_i X_j \notin \mathrm{span}_{\mathbb{Z}_p}[G(X) - G(\boldsymbol{x})]$.

[16] We note that assuming semi-regularity of uniformly random polynomials is a wide-spread heuristic in algebraic cryptanalysis.

[17] In other words, the degree of regularity lower bounds the so-called *first-fall degree*.

A.3.3 A Heuristic Reduction.

We will close the cryptanalytic discussion on MQ+ by a heuristic reduction argument. While we cannot formally prove the correctness of our reduction, it conveys our intuition that the hardness of MQ+ is comparable to the hardness of the well-studied MQ problem. Our basic idea is to reduce MQ to MQ+. I.e., given a solving algorithm for MQ+, we aim to solve MQ. However, our MQ+-solver does not need to be perfect. Indeed, its success probability might only be by some non-negligible function μ better than $2/p$. Hence, we might receive a lot of incorrect solutions from MQ+. To deal with this, we aim to reduce MQ to MQ+ and LPN, the Learning Parity with Noise problem, over $\mathbb{Z}_p$. This is of course a weaker reduction argument (as LPN might simply be harder than MQ), however, it would still surprise us if MQ+ would be easy and MQ directly reducible to LPN (as LPN does on its own not involve any polynomials or higher-degree algebra).

First, let us formally introduce the LPN problem over $\mathbb{Z}_p$:

Definition 11 (Learning Parity with Noise)**.** Let $m \in \text{poly}(k)$ and let $\mathcal{B}_\tau$ be the Bernoulli distribution that outputs 0 with probability $1-\tau$ and a random element of $\mathbb{Z}_p$ with probability τ.

The LPN problem consists of extracting $\boldsymbol{s}$ from $(\boldsymbol{A}, \boldsymbol{A}\boldsymbol{s}+\boldsymbol{e})$ where we sample $\boldsymbol{s} \leftarrow\!\!\$\ \mathbb{Z}_p^k$, $\boldsymbol{A} \leftarrow\!\!\$\ \mathbb{Z}_p^{m\times k}$ and $\boldsymbol{e} \leftarrow\!\!\$\ \mathcal{B}_\tau^m$.

Remark 2. *We note that* LPN *over* $\mathbb{Z}_p$ *with error-rate* τ *is indeed information-theoretically solvable if* τ *is by a noticeable amount smaller than* $1-1/p$ *and* $m \in \text{poly}(k)$ *large enough.*

To see this, we can consider an algorithm that receives m *samples* $(\boldsymbol{A}, \boldsymbol{A}\boldsymbol{s}+\boldsymbol{e}) \in \mathbb{Z}_p^{m\times(k+1)}$ *of* LPN *and divides them in two sets:* $(\boldsymbol{A}_1, \boldsymbol{A}_1\boldsymbol{s}+\boldsymbol{e}_1) \in \mathbb{Z}_p^{m_1\times(k+1)}$ *and* $(\boldsymbol{A}_2, \boldsymbol{A}_2\boldsymbol{s}+\boldsymbol{e}_2) \in \mathbb{Z}_p^{m_2\times(k+1)}$*. It uses the first set* $(\boldsymbol{A}_1, \boldsymbol{A}_1\boldsymbol{s}+\boldsymbol{e}_1)$ *to compute a list of candidate solutions* $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_L$ *for* $(\boldsymbol{A}, \boldsymbol{A}\boldsymbol{s}+\boldsymbol{e})$*. Then, it uses* $(\boldsymbol{A}_2, \boldsymbol{A}_2\boldsymbol{s}+\boldsymbol{e}_2)$ *to filter the list of candidate solutions. The correct solution* $\boldsymbol{s}$ *is expected to satisfy* $(1-\tau)\cdot m_2$ *linear equations of* $\boldsymbol{A}_2$*. However, for a wrong candidate* $\boldsymbol{s}_i$*,* $\boldsymbol{A}_2(\boldsymbol{s}-\boldsymbol{s}_i)+\boldsymbol{e}_2$ *will be uniformly random, resulting in expectedly* m_2/p *equations of* $\boldsymbol{A}_2$ *satisfied. Since* $1-\tau$ *is noticably smaller than* $1/p$*, we can set* $m_2 \in \text{poly}(k)$ *s.t. the correct solution* $\boldsymbol{s}$ *will not get filtered out with overwhelming probability, while each incorrect solution will get filtered with probability at least* $1/2$*.*

Hence, after filtering the list of solutions once, the expected number of incorrect solutions should be at least halved. By sampling more fresh LPN*-samples* $(\boldsymbol{A}_3, \boldsymbol{A}_3\boldsymbol{s}+\boldsymbol{e}_3), (\boldsymbol{A}_4, \boldsymbol{A}_4\boldsymbol{s}+\boldsymbol{e}_4), \ldots$*, we can repeat the filtering step polynomially many times. By Markov's bound, this suffices to reduce the number of incorrect solutions to zero with overwhelming probability.*

Hence, for a computationally unbounded algorithm it is possible to solve LPN *over* $\mathbb{Z}_p$*, even if the error-rate* τ *is only by a non-negligible amount smaller than* $1-1/p$*.*

Now, let us sketch the reduction of MQ to MQ+ and LPN over $\mathbb{Z}_p$ (all problems considered in the *average-case*). We assume that we have a solver for MQ+ with success probability $\frac{2}{p}+\alpha$ for a noticeable α. Additionally, we make the heuristic assumption that the polynomials outputted by MQ+ are random and statistically independent of each other.[18] We assume further, that we have an LPN-oracle that can solve LPN instances

[18] Essentially, we assume that our MQ+-solver outputs a uniformly random degree-2 polynomial $h(X)$ with probability $1-\frac{2}{p}-\alpha$ (i.e. whenever it errs), and a uniformly random degree-2 polynomial $h'(X)$ conditioned on $h'(\boldsymbol{x})=0$ with probability $\frac{2}{p}+\alpha$ (i.e. whenever it computes a correct solution).

with code-dimension $k = \binom{n+2}{2} - 1$ and error-rate $\tau = 1 - \frac{2}{p} - \alpha$ for some $m \in \mathrm{poly}(n)$ large enough.

Given an MQ-instance (G_0, G_1) and $(G_0(\boldsymbol{x}), G_1(\boldsymbol{x}))$, our idea is to query the MQ+-oracle on it multiple times and obtain many polynomials $h_1, \ldots, h_m \in \mathbb{Z}_p[X]^{\leq 2}$. Each h_i vanishes on $\boldsymbol{x}$ with probability $\frac{2}{p} + \alpha$. Denote by $\boldsymbol{s} \in \mathbb{Z}_p^k$ the vector that contains all entries and all quadratic products of entries of $\boldsymbol{x} \in \mathbb{Z}_p^n$. We can parse the noisy quadratic system $h_1(\boldsymbol{x}) = \cdots = h_m(\boldsymbol{x}) = 0$ as a noisy linear equation system $\boldsymbol{A}\boldsymbol{s} = \boldsymbol{b}$, where the i-th row of $\boldsymbol{A}$ contains all coefficients of h_i, except the constant term, and the i-th element of $\boldsymbol{b}$ equals $-h_i(0)$. The error-rate of $\boldsymbol{A}\boldsymbol{s} = \boldsymbol{b}$ is exactly $1 - \frac{2}{p} - \alpha$. Since all coefficients of each h_i are uniformly random and independent (except the constant term), the matrix $\boldsymbol{A}$ is uniformly random too. If we consider the linear system $\boldsymbol{A}\boldsymbol{s} + \boldsymbol{e} = \boldsymbol{b}$ (now with explicit error-vector $\boldsymbol{e} \leftarrow \mathcal{B}^m_{1-\frac{2}{p}-\alpha}$), we see that the secret vector $\boldsymbol{s}$ does not need to be uniformly random. However, it is possible to rerandomize $\boldsymbol{s}$, by sampling $\boldsymbol{s}' \leftarrow\!\!\$\ \mathbb{Z}_p^k$ and considering $\boldsymbol{A}(\boldsymbol{s} + \boldsymbol{s}') + \boldsymbol{e} = \boldsymbol{b} + \boldsymbol{A}\boldsymbol{s}'$. $(\boldsymbol{A}, \boldsymbol{b} + \boldsymbol{A}\boldsymbol{s}')$ is now a proper LPN-instance. If we give it to our LPN-oracle, it will reply with $\boldsymbol{s} + \boldsymbol{s}'$, from which we can subtract $\boldsymbol{s}'$ to get $\boldsymbol{s}$. From $\boldsymbol{s}$, we can deduce all coordinates of $\boldsymbol{x}$, which is the solution of our MQ problem.

Of course, this reduction is very heuristic, as we assume that our MQ+-solver outputs random and statistically independent solutions to our queries. Additionally, it might simply be possible that LPN in this parameter regime is so hard that one can directly reduce MQ to it (without making use of an MQ+-solver). However, this would still surprise us, as there is no obvious reduction of average-case MQ to average-case LPN known.

B On Fast Verification

We will explain here how we can significantly decrease the verification time of the CVRF in § 3 while maintaining security. Currently, Verify needs to evaluate $2n + \ell n\binom{n+2}{2}$ pairings. After optimizations, it will only need to evaluate $(\ell + 1)(n + 1)$ pairings. If we set $n = \ell = 256$, this amounts to 66049 pairings (which would take 1 minute and 6 seconds evaluate, assuming each pairing takes 1 millisecond).

We achieve this by relaxing the unique provability requirement of Def. 5 and allowing for a negligibly small statistical error at verification. This does not go against the spirit of unique provability, as the event of Verify accepting a wrong output image will depend *solely* on the random coins of Verify and is independent of its inputs. Simply put, we randomize Verify such that it checks one random linear combination of quadratic equations per proof step instead of checking n different equations.

Concretely, we will call these relaxed notions *statistical unique provability* and *statistical functionality binding*. They are given by the following:

Definition 12 (Relaxed Unique Provability and Functionality Binding)**.** Let $\mathsf{CVRF}' = (\mathsf{Setup}, \mathsf{Constrain}, \mathsf{Eval}, \mathsf{Verify}', \mathsf{VerifyC}')$ be a tuple of probabilistic algorithms with public parameter space $\mathcal{P} = \mathcal{P}(\lambda)$, input domain $\mathcal{X} = \mathcal{X}(\lambda)$, output codomain $\mathcal{Y} = \mathcal{Y}(\lambda)$, and set of constraints $\mathcal{C} = \mathcal{C}(\lambda) \subseteq 2^{\mathcal{X}}$ where $\emptyset \in \mathcal{C}$. We say that CVRF' is a *relaxed CVRF* if it fulfills d-delegability and (t, q, μ)-Selective Pseudorandomness of Def. 5. Additionally, it has to satisfy:

γ-Statistical Unique Provability: Let $\gamma \in \mathrm{negl}(\lambda)$. For all (even malformed) public

parameters pp, verification keys vk, and proofs π, π' as well as all inputs $x \in \mathcal{X}$ and outputs $y, y' \in \mathcal{Y}$, it holds that

$$y \neq y' \implies \Pr[\mathsf{Verify}'_{\mathsf{pp}}(\mathsf{vk}, x, y, \pi) = \mathsf{Verify}'_{\mathsf{pp}}(\mathsf{vk}, x, y', \pi') = \mathsf{ACCEPT}] \leq \gamma$$

where the probability is taken over the randomness of Verify'.

γ-Statistical Functionality Binding : Let $\gamma \in \mathrm{negl}(\lambda)$. For all (even malformed) public parameters pp, verification keys vk, constraints $C \in \mathcal{C}$, and proofs π_C, π'_C as well as constrained keys $\mathsf{sk}_C, \mathsf{sk}'_C$, it holds that

$$\mathsf{sk}_C \neq \mathsf{sk}'_C \implies \Pr[\mathsf{VerifyC}'_{\mathsf{pp}}(\mathsf{vk}, C, \mathsf{sk}_C, \pi_C) = \mathsf{VerifyC}'_{\mathsf{pp}}(\mathsf{vk}, C, \mathsf{sk}'_C, \pi'_C) = \mathsf{ACCEPT}] \leq \gamma$$

where the probability is taken over the randomness of $\mathsf{VerifyC}'$.[19]

The new relaxed CVRF CVRF' is almost identical with the one given in § 3. The only modified algorithm is $\mathsf{HelperVerify}$ and, subsequently, the verification algorithms Verify and $\mathsf{VerifyC}$. Denote by $\mathbb{Z}_p[X]^{\leq 1}$ and $\mathbb{Z}_p[X]^{\leq 2}$ the space of all polynomials over $X_1, \ldots, X_n$ of degree ≤ 1 and ≤ 2, respectively. Further, Q denotes the space of quadratic maps $G : \mathbb{Z}_p^n \to \mathbb{Z}_p^n$. We detail the new verification algorithms in Fig. 10. To subdivide the analysis of $\mathsf{HelperVerify}$, we introduced three new subroutines $\mathsf{VerifyInit}, \mathsf{VerifyStep}, \mathsf{VerifyFinal}$, presented in Fig. 6.

$\mathsf{HelperVerify}'_{\mathsf{pp}}(\mathsf{vk} \in \mathbb{G}_2^n, x \in \{0,1\}^{\ell'}, \pi_x \in (\mathbb{G}_1^n)^{\ell'}, \tilde{\pi}^{(\ell')} \in \mathbb{G}_1^n)$**:**

01 Parse $\pi_x = (\tilde{\pi}^{(i)})_{i=0}^{\ell'-1}$ and $\mathsf{pp} = ((G_0^{(i)}, G_1^{(i)}))_{i=1}^{\ell}$
02 If $\mathsf{VerifyInit}_{\mathsf{pp}}(\mathsf{vk}, \pi^{(0)}) = \mathsf{REJECT}$, output REJECT
03 For $1 \leq i \leq \ell'$:
04 If $\mathsf{VerifyStep}_{\mathsf{pp}}(G_{x_i}^{(i)}, \pi^{(i-1)}, \pi(i)) = \mathsf{REJECT}$, output REJECT
05 Output ACCEPT

$\mathsf{CVRF}.\mathsf{VerifyC}'_{\mathsf{pp}}(\mathsf{vk} \in \mathbb{G}_2^n, x \in \{0,1\}^{\ell'}, \mathsf{sk}_x \in \mathbb{Z}_p^n, \pi_x \in (\mathbb{G}_1^n)^{\ell'})$**:**

01 If $\ell' \geq \ell$, output REJECT
02 Output $\mathsf{HelperVerify}'_{\mathsf{pp}}(\mathsf{vk}, x, \pi_x, \mathsf{g}_1^{\mathsf{sk}_x})$

$\mathsf{CVRF}.\mathsf{Verify}'_{\mathsf{pp}}(\mathsf{vk} \in \mathbb{G}_2^n, x \in \{0,1\}^{\ell}, \boldsymbol{y} \in \mathbb{G}_2^n, \pi = (\pi_x \in (\mathbb{G}_1^n)^{\ell}, \tilde{\pi}^{(\ell)} \in \mathbb{G}_1^n))$**:**

01 If $\mathsf{HelperVerify}'_{\mathsf{pp}}(\mathsf{vk}, x, \pi_x, \tilde{\pi}^{(\ell)}) = \mathsf{REJECT}$, output REJECT
02 Output $\mathsf{VerifyFinal}(\tilde{\pi}^{(\ell)}, \boldsymbol{y})$

Figure 5: Pseudocode of the new verification algorithms for CVRF'. All other algorithms of CVRF' are given in Fig. 2.

We will now prove formally that CVRF' is a relaxed CVRF:

[19] Note that we simplified the definition of statistical functionality binding in comparison to the more general definition of Def. 5. This notion is stricter, as it requires the equality of constrained keys, while the notion of Def. 5 only required them to be functionally equivalent. We did this purely to simplify the following arguments.

$\mathsf{VerifyInit}(\mathsf{vk} \in \mathbb{G}_2^n, \tilde{\pi} \in \mathbb{G}_1^n)$:

Parse $(\mathsf{g}_2^{a_1}, \ldots, \mathsf{g}_2^{a_n}) = \mathsf{vk}$
Parse $(\mathsf{g}_1^{b_1}, \ldots, \mathsf{g}_1^{b_n}) = \tilde{\pi}$
Draw $\alpha_1, \ldots, \alpha_n \leftarrow_\$ \mathbb{Z}_p$
Compute $\mathtt{u}_2 = \prod_{i=1}^n (\mathsf{g}_2^{a_i})^{\alpha_i} \in \mathbb{G}_2$
Compute $\mathtt{w}_1 = \prod_{i=1}^n \left(\mathsf{g}_1^{b_i}\right)^{\alpha_i} \in \mathbb{G}_1$
If $e(\mathtt{w}_1, \mathsf{g}_1) = \mathtt{u}_2$, output ACCEPT
Otherwise, output REJECT

$\mathsf{VerifyStep}(G \in Q, \tilde{\pi} \in \mathbb{G}_1^n, \tilde{\pi}' \in \mathbb{G}_1^n)$:

If G is not a degree-2 map, output REJECT
Parse $(\mathsf{g}_1^{a_1}, \ldots, \mathsf{g}_1^{a_n}) = \tilde{\pi}$
Parse $(\mathsf{g}_1^{b_1}, \ldots, \mathsf{g}_1^{b_n}) = \tilde{\pi}'$
Parse $(g_1, \ldots, g_n) = G$
Draw $\alpha_1, \ldots, \alpha_n \leftarrow_\$ \mathbb{Z}_p$
Compute $\mathtt{v}_1 = \prod_{i=1}^n \left(\mathsf{g}_1^{b_i}\right)^{\alpha_i}$
Compute $h(X) := \alpha_1 \cdot g_1(X) + \cdots + \alpha_n \cdot g_n(X) \in \mathbb{Z}_p[X]^{\leq 2}$
Compute linear forms $l_0, \cdots, l_n \in \mathbb{Z}_p[X]^{\leq 1}$ s.t. $h(X) = X_1 \cdot l_1(X) + \cdots + X_n \cdot l_n(X) + l_0(X)$
For $0 \leq i \leq n$:
  Parse $l_i(X) = \beta_1 \cdot X_1 + \cdots + \beta_n \cdot X_n + \beta_0$ with $\beta_0, \ldots, \beta_n \in \mathbb{Z}_p$
  Compute $\mathtt{u}_1^{(i)} = \mathsf{g}_1^{\beta_0} \cdot \prod_{i=1}^n (\mathsf{g}_1^{a_i})^{\beta_i}$
For $1 \leq i \leq n$:
  Compute $\mathtt{w}_2^{(i)} = e(\mathtt{u}_1^{(i)}, \mathsf{g}_1^{a_i})$
Compute $\mathtt{w}_2^{(0)} = e(\mathtt{u}_1^{(0)} \cdot \mathtt{v}_1^{-1}, \mathsf{g}_1)$
If $\prod_{i=0}^n \mathtt{w}_2^{(i)} = \mathsf{g}_2^0$, output ACCEPT
Otherwise, output REJECT

$\mathsf{VerifyFinal}(\tilde{\pi} \in \mathbb{G}_1^n, \boldsymbol{y} \in \mathbb{G}_2^n)$:

Parse $(\mathsf{g}_1^{a_1}, \ldots, \mathsf{g}_1^{a_n}) = \tilde{\pi}$
Parse $(\mathsf{g}_2^{b_1}, \ldots, \mathsf{g}_2^{b_n}) = \boldsymbol{y}$
For $1 \leq i \leq n$:
  If $\mathsf{g}_2^{b_i} \neq e(\mathsf{g}_1^{a_i}, \mathsf{g}_1^{a_i})$, output REJECT
Output ACCEPT

Figure 6: Pseudocode of additional auxiliary algorithms for $\mathsf{HelperVerify}$.

Theorem 6. *Let* $\gamma = \frac{\ell+1}{p}$. *Then,* $\mathsf{CVRF}' = (\mathsf{Setup}, \mathsf{Constrain}, \mathsf{Eval}, \mathsf{Verify}', \mathsf{VerifyC}')$ *is a relaxed CVRF in the sense of Def. 12 and fulfills* γ*-statistical unique provability and* γ*-statistical functionality binding.*

The time complexity of Verify' *is dominated by evaluating* $(n+1)(\ell+1)$ *pairings.*

Proof. Note that delegability, constraint-hiding and pseudorandomness for $\mathsf{CVRF}' =$

$(\mathsf{Setup}, \mathsf{Constrain}, \mathsf{Eval}, \mathsf{Verify}', \mathsf{VerifyC}')$ with the modified verification algorithms follow from Thms. 3 and 4, as those properties are independent of Verify and $\mathsf{VerifyC}$.

We will prove here that CVRF' is γ-statistical unique provable. (The proof for γ-statistical functionality binding works analogously.) For this end, let $\mathsf{vk} = \mathsf{g}_2^{\boldsymbol{s}}$ for some $\boldsymbol{s} \in \mathbb{Z}_p^n$. Let $G^{(i)} = (G_0^{(i)}, G_1^{(i)}) \in Q^2$ for $i \in [\ell]$. Let $\tilde{\pi}^{(i)} \in \mathbb{G}_1^n$ for $i \in [0, \ell]$. Finally, let $x \in \{0,1\}^\ell$ and $\boldsymbol{y} \in \mathbb{G}_2^n$. We have to prove that $\mathsf{Verify}'_{\mathsf{pp}}$ rejects $(\mathsf{vk}, x, \boldsymbol{y}, (\tilde{\pi}^{(i)})_{i=0}^{\ell}))$ with probability at least $1 - \gamma$ whenever $\boldsymbol{y} \neq \mathsf{g}_2^{G_x(\boldsymbol{s})^2}$.

For this end, it suffices to analyze $\mathsf{VerifyInit}$, $\mathsf{VerifyStep}$ and $\mathsf{VerifyFinal}$:

1. On input $\mathsf{vk} = \mathsf{g}_2^{\boldsymbol{a}}$ and $\widetilde{\pi} = \mathsf{g}_1^{\boldsymbol{b}}$, $\mathsf{VerifyInit}$ draws a uniformly random vector $\alpha \leftarrow_\$ \mathbb{Z}_p^n$ and computes the inner products $\langle \boldsymbol{a}, \alpha \rangle$ and $\langle \boldsymbol{b}, \alpha \rangle$ in the exponents of $\mathtt{u}_2$ and $\mathtt{w}_1$. It uses one pairing to verify that we have

$$\langle \boldsymbol{a}, \alpha \rangle = \langle \boldsymbol{b}, \alpha \rangle.$$

   If $\boldsymbol{a} = \boldsymbol{b}$, then the above equality will always hold. On the other hand, if $\boldsymbol{a} \neq \boldsymbol{b}$, then, by a Schwartz-Zippel argument, the above equality will not hold with probability $1 - 1/p$.

   Hence, $\mathsf{VerifyInit}$ will reject inputs vk, $\widetilde{\pi}$ with different exponents (with respect to g_2 and g_1) with overwhelming probability $1 - 1/p$.

2. On input $G \in Q$, $\widetilde{\pi} = \mathsf{g}_1^{\boldsymbol{a}}$ and $\widetilde{\pi}' = \mathsf{g}_1^{\boldsymbol{b}}$, we claim that $\mathsf{VerifyStep}$ always accepts if $\boldsymbol{b} = G(\boldsymbol{a})$. On the other hand, it rejects with probability $1 - 1/p$ if $\boldsymbol{b} \neq G(\boldsymbol{a})$.

   At start, $\mathsf{VerifyStep}$ samples $\alpha_1, \ldots, \alpha_n \leftarrow_\$ \mathbb{Z}_p^n$ and sets

$$h(X) := \alpha_1 \cdot g_1(X) + \cdots + \alpha_n \cdot g_n(X).$$

   Additionally, it computes $\langle \alpha, \boldsymbol{b} \rangle$ in the exponent of $\mathtt{v}_1 = \mathsf{g}_1^{\langle \alpha, \boldsymbol{b} \rangle}$. As next step, it computes linear forms $l_0, \ldots, l_n \in \mathbb{Z}_p[X]^{\leq 1}$ such that

$$h(X) = X_1 \cdot l_1(X) + \cdots + X_n \cdot l_n(X) + l_0(X).$$

   Note that there is no unique choice for such linear forms, however, if $h(X) = \sum_{1 \leq i \leq j \leq n} c_{i,j} X_i X_j + \sum_{i=1}^n d_i X_i + d_0$, a straightforward approach is given by

$$\begin{aligned}
l_1(X) &= \sum_{i=1}^{n} c_{1,i} X_i, \\
l_2(X) &= \sum_{i=2}^{n} c_{2,i} X_i, \\
&\vdots \\
l_n(X) &= c_{n,n} X_n, \\
l_0(X) &= \sum_{i=1}^{n} d_i X_i + d_0.
\end{aligned}$$

Whatever choices for $l_0, \ldots, l_n$ are made by $\mathsf{VerifyStep}$, in the next step it computes the values $l_i(\boldsymbol{a})$ in the exponents of $\mathtt{u}_i = \mathsf{g}_1^{l_i(\boldsymbol{a})}$, for $i \in [0, n]$. As next step, it uses n pairings to compute

$$\mathtt{w}_2^{(i)} = e(\mathtt{u}_1^{(i)}, \mathsf{g}_1^{a_i}) = \mathsf{g}_2^{a_i \cdot l_i(\boldsymbol{a})}$$

for $i \in [n]$. Finally, it uses one more pairing to set

$$\mathtt{w}_2^{(0)} = e(\mathtt{u}_1^{(0)} \cdot \mathtt{v}_1^{-1}, \mathsf{g}_1) = \mathsf{g}_2^{l_0(\boldsymbol{a}) - \langle \alpha, \boldsymbol{b} \rangle}.$$

$\mathsf{VerifyStep}$ accepts if $\prod_{i=0}^{n} \mathtt{w}_2^{(i)}$ is the neutral element of the group. This is the case iff

$$\begin{aligned} & \left(\sum_{i=1}^{n} a_i \cdot l_i(\boldsymbol{a}) \right) + l_0(\boldsymbol{a}) - \langle \alpha, \boldsymbol{b} \rangle = 0 \\ \iff & l_0(\boldsymbol{a}) + a_1 \cdot l_1(\boldsymbol{a}) + \cdots + a_n \cdot l_n(\boldsymbol{a}) = \langle \alpha, \boldsymbol{b} \rangle \\ \iff & h(\boldsymbol{a}) = \langle \alpha, \boldsymbol{b} \rangle \\ \iff & \langle \alpha, G(\boldsymbol{a}) \rangle = \langle \alpha, \boldsymbol{b} \rangle. \end{aligned}$$

If $G(\boldsymbol{a}) = \boldsymbol{b}$, then the above inequality will always hold. On the other hand, if $G(\boldsymbol{a}) \neq \boldsymbol{b}$, then the equality $\langle \alpha, G(\boldsymbol{a}) \rangle = \langle \alpha, \boldsymbol{b} \rangle$ will only hold with probability $1/p$ over the randomness of α.

3. On input $\widetilde{\pi} = \mathsf{g}_1^{\boldsymbol{a}}$ and $\boldsymbol{y} = \mathsf{g}_2^{\boldsymbol{b}}$, $\mathsf{VerifyFinal}$ uses n pairings to verify that each coordinate of $\boldsymbol{b}$ is the square of the corresponding coordinate of $\boldsymbol{a}$. $\mathsf{VerifyFinal}$ accepts if and only if $\boldsymbol{b} = \boldsymbol{a}^2$.

In total, $\mathsf{VerifyInit}$ and $\mathsf{VerifyStep}$ incur a statistical error of $1/p$ per execution, while $\mathsf{VerifyFinal}$ is perfectly correct. As Verify' induces ℓ calls to $\mathsf{VerifyStep}$ and one call to $\mathsf{VerifyInit}$, by a union bound its statistical error is bounded by $\gamma = \frac{\ell+1}{p}$. Hence, CVRF' is γ-statistical unique provable.

Further, the number of pairings executed of Verify' is given by the number of parings used by $\mathsf{VerifyInit}$, $\mathsf{VerifyFinal}$ and ℓ times the number of pairings used by $\mathsf{VerifyStep}$. Hence, Verify' needs $1 + n + \ell \cdot (n+1) = (\ell+1)(n+1)$ pairings. □

C Constrained VRFs from Generic Assumptions

In this section, we extend the generic approach for VRFs based on non-interactive witness-indistinguishability proofs (NIWI) [GHKW17, Bit20] to *constrained* VRFs. We stress that this construction has two major disadvantages in comparison to our pairing-based construction of § 3: first, it is not constraint-hiding in the sense of Def. 6. Hence, its security in the selective pseudorandomness game does not model the capabilities of a passive adversary in full complexity as explained in Remark 1. Second, it is not practical mainly due to the large size of the involved NIWI proofs.

First, we recall the original[20] construction of [GHKW17, Bit20]. We present a version that is only selectively secure (omitting the partitioning scheme for adaptive security).

[20] To keep the paper concise, we use the notions of commitment schemes, non-interactive witness-indistinguishable proofs and (normal) verifiable random functions without formal introduction. For definitions, we refer to, e.g., [Bit20].

Construction 1 (NIWI-Based VRF [GHKW17, Bit20])**.** Let COM be a non-interactive commitment scheme. Let NIWI be a NIWI. Let PPRF be a puncturable PRF[21]. We present the VRF VRF of [GHKW17, Bit20] in Fig. 7 using the statement

$$\mathsf{stmt}_{\mathsf{vk},x,y} := \left\{ \begin{array}{c} \exists i \neq j \in \{1,2,3\}, \widetilde{k}_i, \widetilde{k}_j, \widetilde{r}_i, \widetilde{r}_j : \\ y = \mathsf{PPRF.Eval}(\widetilde{k}_i; x) = \mathsf{PPRF.Eval}(\widetilde{k}_j; x) \\ \wedge\ c_i = \mathsf{COM.Com}(\widetilde{k}_i; \widetilde{r}_i) \wedge\ \ c_j = \mathsf{COM.Com}(\widetilde{k}_j; \widetilde{r}_j) \end{array} \right\} . \tag{2}$$

$\mathsf{VRF.Setup}(1^\lambda)$**:**

```
01 k ← PPRF.Gen(1^λ)
02 r_1, r_2, r_3 ← {0,1}^λ
03 c_1 := COM.Com(k; r_1)
04 c_2 := COM.Com(k; r_2)
05 c_3 := COM.Com(k; r_3)
06 vk := (c_1, c_2, c_3)
07 sk := (k, r_2, r_3, vk)
08 return (sk, vk)
```

$\mathsf{VRF.Eval}(\mathsf{sk}, x)$**:**

```
01 (k, r_2, r_3, vk) := sk
02 y := PPRF.Eval(k, x)
03 w := (2, 3, k, k, r_2, r_3)
04 π ← NIWI.Prove(stmt_{vk,x,y}, w)
05 return (y, π)
```

$\mathsf{VRF.Verify}(\mathsf{vk}, x, y, \pi)$**:**

```
01 return NIWI.Verify(stmt_{vk,x,y}, π)
```

Figure 7: NIWI-based VRF.

The trick here is that the secret key k that allows evaluation of the PPRF is committed to three times and whenever an evaluation needs to be proven correct, it is shown that the evaluation is consistent with at least two out of the three committed keys. Given the perfect binding of the underlying commitment and the perfect soundness of NIWI proofs, this implies that there can never be two distinct provable outputs for any given input, as a majority of the committed keys need to agree with the output.

Towards security, we explain how Bitansky [Bit20] achieved pseudorandomness in the *selective* security setting. Once we know the challenge point x^*, the key k is carefully replaced by a key k_{x^*}, which is punctured at that point. To facilitate this, we notice that the first commitment is not used at all in the honest NIWI witness, so we can switch c_1

[21] By *puncturable PRF*, we mean a CPRF which only needs to support the set of constraints $\mathcal{C} = \{\{x\} | x \in \mathcal{X}\}$. I.e., it punctures out single points.

to a commitment to k_{x^*} due to the hiding property of the commitment scheme. Then, step-by-step, we apply witness indistinguishability of the NIWIs to switch all witnesses to the commitments c_1, c_2 instead of c_2, c_3. This works because we evaluate only on points that are not x^*, so the evaluation is still consistent with the punctured key committed to by c_1. Similarly, we repeat this until all commitments have been swapped to the key k_{x^*}. From there, if the adversary could distinguish $\mathsf{PPRF.Eval}(k, x)$ from random, it can break the pseudorandomness game of the underlying CPRF.

Now, if we want to build a CVRF for constraint class $\mathcal{C}$, we replace the puncturable PRF PPRF by a puncturable constrained PRF CPRF, which allows puncturing out single points, but also more expressive constraints in $\mathcal{C}$. In principle, we can repeat the same logic we applied to Eval to the constraining algorithm. However, looking, ahead there will be some caveats. We call the root secret key k^0 and its verification key $\mathsf{vk} = (c_1^0, c_2^0, c_3^0)$ consists of commitments as above. When we constrain k^0 with constraint C, we invoke constraining of the underlying CPRF to get $k^1 = \mathsf{CPRF.Constrain}(k^0, C)$ and additionally create a new commitment triplet (c_1^1, c_1^2, c_1^3) of k^1. The triplet (c_1^1, c_1^2, c_1^3) can be seen as a constrained version of the public key. Then, we give a proof π that at least two of the new commitments c_i^1 contain keys that are directly received by constraining at least two of the keys included in the base commitments c_i^0 with constraint C in a NIWI fashion. In this way, by the functionality preserving property of the underlying CPRF, we know that at least two keys in the new commitment must be functionally equivalent.

The full constrained key consists of k^1, the commitments c_i^1, the randomness r_2^1, r_3^1 used to generate two of those commitments and the proof π. The receiver of these elements can verify that c_2^1, c_3^1 indeed commit to k^1 and k^1 is received by constraining at least two of the keys in the original vk.

Further, the structure of the constrained key plus these proof elements is essentially of the same shape as holding a root key and commitments with associated randomness in the base scheme. Thus, it allows the receiver to either further constrain k^1 in the same manner or evaluate on any point which is admissible for k^1 while also providing a proof in the same way as outlined above.

For example, to show that a further constrained key $k^2 = \mathsf{CPRF.Constrain}(k^1, C')$ with commitments c_1^2, c_2^2, c_3^2 was generated correctly, one would hand out vk and the intermediate commitments (c_1^1, c_2^1, c_3^1), and show that two of $\{c_1^1, c_2^1, c_3^1\}$ are correctly derived with respect to vk and constraint C and, then, show that two of $\{c_1^2, c_2^2, c_3^2\}$ are correctly derived with respect to (c_1^1, c_2^1, c_3^1) and constraint C'.

It is obvious, that this simple construction is *not constraint-hiding*, as the history of constraints is explicitly needed to verify the chain of proofs necessary to verify a multiply constrained key. Moreover, this problem seems to be somewhat inherent with our general approach of a layered proof system, as this intuitively leaks information at least about the number of times that a key was punctured before. There are also some caveats concerning a meaningful security definition for a CVRF that is not constraint-hiding, which will also be discussed in Remark 3.

Let us now present the augmented construction with a constraining mechanism. To simplify the presentation, we assume that the $\mathsf{CPRF.Constrain_{pp}}$ algorithm is deterministic.

Construction 2 (NIWI-Based CVRF)**.** Let COM be a non-interactive commitment scheme. Let NIWI be a NIWI. Let CPRF be a puncturable constrained PRF. We present the CVRF CVRF based on the VRF of [GHKW17, Bit20] in Figs. 8 and 9. It uses the

statements

$$\mathsf{stmt}_{c_1,c_2,c_3,x,y} := \left\{ \begin{array}{c} \exists i \neq j \in \{1,2,3\}, \widetilde{k}_i, \widetilde{k}_j, \widetilde{r}_i, \widetilde{r}_j : \\ y = \mathsf{CPRF.Eval}(\widetilde{k}_i; x) = \mathsf{CPRF.Eval}(\widetilde{k}_j; x) \\ \wedge\ c_i = \mathsf{COM.Com}(\widetilde{k}_i; \widetilde{r}_i) \wedge\ \ c_j = \mathsf{COM.Com}(\widetilde{k}_j; \widetilde{r}_j) \end{array} \right\} \tag{3}$$

$$\mathsf{stmt}_{C,c_1,c_2,c_3,c'_1,c'_2,c'_3} := \left\{ \begin{array}{c} \exists i \neq j \in \{1,2,3\}, \widetilde{k}_i, \widetilde{k}_j, \widetilde{r}_i, \widetilde{r}_j, \widetilde{k}', \widetilde{r}'_i, \widetilde{r}'_j : \\ \widetilde{k}' = \mathsf{CPRF.Constrain}(\widetilde{k}_i, C) = \mathsf{CPRF.Constrain}(\widetilde{k}_j, C) \\ \wedge\ c_i = \mathsf{COM.Com}(\widetilde{k}_i; \widetilde{r}_i) \wedge\ \ c_j = \mathsf{COM.Com}(\widetilde{k}_j; \widetilde{r}_j) \\ \wedge\ c'_i = \mathsf{COM.Com}(\widetilde{k}'; \widetilde{r}'_i) \wedge\ \ c'_j = \mathsf{COM.Com}(\widetilde{k}'; \widetilde{r}'_j) \end{array} \right\} . \tag{4}$$

Now, we formally prove that Construction 2 is selectively secure, delegable, and uniquely provable. As a minor technical detail, we need to require that the underlying CPRF is *puncture invariant*:

Definition 13 (Puncture Invariance)**.** A 2-delegable puncturable CPRF with set of constraints $\mathcal{C}$ is called *puncture invariant* if we have for all $C \in \mathcal{C}$ and $x \in C$

$$\mathsf{Constrain}(\mathsf{sk}_\emptyset, C) = \mathsf{Constrain}(\mathsf{sk}_{\{x\}}, C) \tag{5}$$

where $\mathsf{sk}_\emptyset \leftarrow \mathsf{Setup}(1^\lambda)$ and $\mathsf{sk}_{\{x\}} \leftarrow \mathsf{Constrain}(\mathsf{sk}_\emptyset, \{x\})$.

Theorem 7. *Let* COM *be a* $(t, \epsilon_{\mathsf{COM}})$*-hiding non-interactive commitment scheme. Let* NIWI *be a* $(t, \epsilon_{\mathsf{NIWI}})$*-indistinguishable NIWI. Let* CPRF *be a puncturable, 2-delegable, puncture-invariant, selectively* $(t, \epsilon_{\mathsf{CPRF}})$*-pseudorandom CPRF. Then, for every* q*, the CVRF* CVRF *in Construction 2 is selectively* $(t - t', q, \epsilon)$*-pseudorandom, where*

$$t' = \mathrm{poly}(\lambda, q) \tag{6}$$

$$\epsilon(\lambda) = 6 \cdot \epsilon_{\mathsf{COM}}(\lambda) + 4 \cdot q(\lambda) \cdot \epsilon_{\mathsf{NIWI}}(\lambda) + \epsilon_{\mathsf{CPRF}}(\lambda) \tag{7}$$

for some sufficiently large polynomial poly*.*

Proof. We prove the pseudorandomness of the CVRF CVRF via a sequence of hybrid games. The proof essentially follows the strategy of [GHKW17, Bit20].

Game 0 **(Original game with** $b = 0$**):** This is the original game where the CVRF CVRF is run as described in Construction 2 with challenge bit $b = 0$, i.e., $(y^*, \pi^*) = \mathsf{CVRF.Eval}(\mathsf{sk}, x^*)$. In particular, the adversary has to first supply the challenger with the challenge preimage x^*. Recall that $k^0 \leftarrow \mathsf{CPRF.Setup}(1^\lambda)$ is the CPRF key of the underlying CPRF.

Game 1 **(Switch the first commitment to a punctured key):** In this game, the first commitment is generated as $c_1^0 := \mathsf{COM.Com}(k_{x^*}; r_1^0)$ where $k_{x^*} := \mathsf{CPRF.Constrain}(k^0, \{x^*\})$ instead of $c_1^0 := \mathsf{COM.Com}(k^0; r_1^0)$. The distinguishing advantage of the adversary between this game and the previous is at most

$$|\Pr[\text{Game 1}] - \Pr[\text{Game 0}]| \leq \epsilon_{\mathsf{COM}}(\lambda) \tag{8}$$

because the randomness r_1^0 is not used in the NIWI proofs.

$\mathsf{CVRF.Setup}(1^\lambda)$:

$k^0 \leftarrow \mathsf{CPRF.Gen}(1^\lambda)$
$r_1^0, r_2^0, r_3^0 \leftarrow \{0,1\}^\lambda$
$c_1^0 \coloneqq \mathsf{COM.Com}(k^0; r_1)$, $c_2^0 \coloneqq \mathsf{COM.Com}(k^0; r_2)$, and $c_3^0 \coloneqq \mathsf{COM.Com}(k^0; r_3)$
$\mathsf{vk} \coloneqq (c_1^0, c_2^0, c_3^0)$
$\mathsf{sk}_\emptyset \coloneqq k^0$, $\mathsf{pp} \coloneqq \varepsilon$, and $\pi_\emptyset \coloneqq (r_2^0, r_3^0, \varepsilon)$
return $(\mathsf{pp}, \mathsf{sk}_\emptyset, \mathsf{vk}, \pi_\emptyset)$

$\mathsf{CVRF.Constrain}(\mathsf{sk}, \pi, C \in \mathcal{C})$:

$k^j \coloneqq \mathsf{sk}$
$(r_2^j, r_3^j, (C^i, c_1^i, c_2^i, c_3^i, \pi^i)_{i\in[j]}) \coloneqq \pi$
$k^{j+1} \coloneqq \mathsf{CPRF.Constrain}(k^j, C)$
$r_1^{j+1}, r_2^{j+1}, r_3^{j+1} \leftarrow \{0,1\}^\lambda$
$c_1^{j+1} \coloneqq \mathsf{COM.Com}(k^{j+1}; r_1^{j+1})$, $c_2^{j+1} \coloneqq \mathsf{COM.Com}(k^{j+1}; r_2^{j+1})$, and $c_3^{j+1} \coloneqq \mathsf{COM.Com}(k^{j+1}; r_3^{j+1})$
$\mathsf{w}^{j+1} \coloneqq (2, 3, k^j, k^j, r_2^j, r_3^j, k^{j+1}, r_2^{j+1}, r_3^{j+1})$
$C^{j+1} \coloneqq C$
$\pi^{j+1} \leftarrow \mathsf{NIWI.Prove}(\mathsf{stmt}_{C^{j+1}, c_1^j, c_2^j, c_3^j, c_1^{j+1}, c_2^{j+1}, c_3^{j+1}}, \mathsf{w}^{j+1})$
$\pi' \coloneqq (r_2^{j+1}, r_3^{j+1}, (C^i, c_1^i, c_2^i, c_3^i, \pi^i)_{i\in[j+1]})$
$\mathsf{sk}' \coloneqq k^{j+1}$
return (sk', π')

$\mathsf{CVRF.Eval}(\mathsf{sk}, \pi, x)$:

$k^j \coloneqq \mathsf{sk}$
$(r_2^j, r_3^j, (C^i, c_1^i, c_2^i, c_3^i, \pi^i)_{i\in[j]}) \coloneqq \pi$
$y \coloneqq \mathsf{CPRF.Eval}(k^j, x)$
$\mathsf{w}_x \coloneqq (2, 3, k^j, k^j, r_2^j, r_3^j)$
$\pi_x \leftarrow \mathsf{NIWI.Prove}(\mathsf{stmt}_{c_1^j, c_2^j, c_3^j, x, y}, \mathsf{w}_x)$
$\pi' \coloneqq (\pi_x, (C^i, c_1^i, c_2^i, c_3^i, \pi^i)_{i\in[j]})$
return (y, π')

Figure 8: NIWI-based CVRF, evaluation algorithms.

$\mathsf{CVRF.Verify}(\mathsf{vk}, x, y, \pi)$:

01 $(c_1^0, c_2^0, c_3^0) \coloneqq \mathsf{vk}$
02 $(\pi_x, (C^i, c_1^i, c_2^i, c_3^i, \pi^i)_{i\in[j]}) \coloneqq \pi$
03 require $x \notin C^j$
04 for all $i \in \{1, ..., j-1\}$
05 require $C_i \subseteq C_{i+1}$
06 for all $i \in \{0, ..., j-1\}$
07 require $\mathsf{NIWI.Verify}(\mathsf{stmt}_{C^{i+1},c_1^i,c_2^i,c_3^i,c_1^{i+1},c_2^{i+1},c_3^{i+1}}, \pi^{i+1}) = 1$
08 return $\mathsf{NIWI.Verify}(\mathsf{stmt}_{c_1^j,c_2^j,c_3^j,x,y}, \pi_x)$

$\mathsf{CVRF.VerifyC}(\mathsf{vk}, C, \mathsf{sk}', \pi')$:

01 $(c_1^0, c_2^0, c_3^0) \coloneqq \mathsf{vk}$
02 $k^{j+1} \coloneqq \mathsf{sk}'$
03 $(r_2^{j+1}, r_3^{j+1}, (C^i, c_1^i, c_2^i, c_3^i, \pi^i)_{i\in[j+1]}) \coloneqq \pi'$
04 require $c_2^{j+1} = \mathsf{COM.Com}(k^{j+1}; r_2^{j+1}) \wedge c_3^{j+1} = \mathsf{COM.Com}(k^{j+1}; r_3^{j+1})$
05 for all $i \in \{1, ..., j-1\}$
06 require $C_i \subseteq C_{i+1}$
07 for all $i \in \{0, ..., j-1\}$
08 require $\mathsf{NIWI.Verify}(\mathsf{stmt}_{C^{i+1},c_1^i,c_2^i,c_3^i,c_1^{i+1},c_2^{i+1},c_3^{i+1}}, \pi^{i+1}) = 1$
09 return $C_j = C$

Figure 9: NIWI-based CVRF, verification algorithms.

Game $2.i$ **(Switch evaluation proof witnesses for** $i \in [0, q_2(\lambda)]$**):** For each evaluation in query $\iota < i$ with preimage x_ι, we use the witness $\mathsf{w}_x := (1, 3, k_{x^*}, k, r_1^0, r_3^0)$ instead of $\mathsf{w}_x := (2, 3, k, k, r_2^0, r_3^0)$. Note that both witnesses are valid for the statement $\mathsf{stmt}_{c_1^0, c_2^0, c_3^0, x, y}$ due to the 2-delegability of CPRF. This is analogous to the original proof in [GHKW17, Bit20]. Consequently, the distinguishing advantage of the adversary between this game and the previous is at most

$$|\Pr[\text{Game } 2.i] - \Pr[\text{Game } 2.(i-1)]| \leq \epsilon_{\mathsf{NIWI}}(\lambda) \tag{9}$$

by the witness indistinguishability of NIWI. Overall, we find

$$|\Pr[\text{Game } 2.0] - \Pr[\text{Game } 2.q_2(\lambda)]| \leq q_2(\lambda) \cdot \epsilon_{\mathsf{NIWI}}(\lambda) \ . \tag{10}$$

Moreover, Game 2.0 is the same game as Game 1.

Game $3.j$ **(Switch constrained proof witnesses for** $j \in [0, q_1(\lambda)]$**):** For each constrain query $\jmath < j$ with constraint $C_\jmath$, we use the witness $\mathsf{w}_{C_\jmath} := (1, 3, k_{x^*}, k^0, r_1^0, r_3^0, k^1, r_1^1, r_3^1)$ instead of $\mathsf{w}_{C_\jmath} := (2, 3, k^0, k^0, r_2^0, r_3^0, k^1, r_1^1, r_3^1)$ where $k^1 := \mathsf{CPRF.Constrain}(k^0, C_\jmath)$. Note that both witnesses are valid for the statement $\mathsf{stmt}_{C_\jmath, c_1^0, c_2^0, c_3^0, c_1^1, c_2^1, c_3^1}$. This follows crucially from the puncture invariance of CPRF because $x^* \in C_\jmath \implies \mathsf{CPRF.Constrain}(k^0, C_\jmath) = \mathsf{CPRF.Constrain}(k_{x^*}, C_\jmath)$. Consequently, the distinguishing advantage of the adversary between this game and the previous is at most

$$|\Pr[\text{Game } 3.j] - \Pr[\text{Game } 3.(j-1)]| \leq \epsilon_{\mathsf{NIWI}}(\lambda) \tag{11}$$

by the witness indistinguishability of NIWI. Overall, we find

$$|\Pr[\text{Game } 3.0] - \Pr[\text{Game } 3.q_1(\lambda)]| \leq q_1(\lambda) \cdot \epsilon_{\mathsf{NIWI}}(\lambda) \ . \tag{12}$$

Moreover, Game 3.0 is the same game as Game $2.q_2(\lambda)$.

Game 4 **(Switch the second commitment to a punctured key):** In this game the second commitment is generated as $c_2^0 := \mathsf{COM.Com}(k_{x^*}; r_2^0)$ where $k_{x^*} := \mathsf{CPRF.Constrain}(k^0, \{x^*\})$ instead of $c_2^0 := \mathsf{COM.Com}(k^0; r_2^0)$. The distinguishing advantage of the adversary between this game and the previous is at most

$$|\Pr[\text{Game } 4] - \Pr[\text{Game } 3.q_1(\lambda)]| \leq \epsilon_{\mathsf{COM}}(\lambda) \tag{13}$$

because the randomness r_2^0 is not used in the NIWI proofs.

Game $5.i$ **(Switch evaluation proof witnesses for** $k \in [0, q_2(\lambda)]$**):** For each evaluation query $\iota < i$ with preimage x_ι, we use the witness $\mathsf{w} := (1, 2, k_{x^*}, k_{x^*}, r_1^0, r_2^0)$ instead of $\mathsf{w} := (1, 3, k_{x^*}, k^0, r_1^0, r_3^0)$. Note that both witnesses are valid for the statement $\mathsf{stmt}_{c_1^0, c_2^0, c_3^0, x, y}$ due to the puncture invariance of CPRF. Consequently, the distinguishing advantage of the adversary between this game and the previous is at most

$$|\Pr[\text{Game } 5.i] - \Pr[\text{Game } 5.(i-1)]| \leq \epsilon_{\mathsf{NIWI}}(\lambda) \tag{14}$$

by the witness indistinguishability of NIWI. Overall, we find

$$|\Pr[\text{Game } 5.0] - \Pr[\text{Game } 5.q_2(\lambda)]| \leq q_2(\lambda) \cdot \epsilon_{\mathsf{NIWI}}(\lambda) \ . \tag{15}$$

Moreover, Game 5.0 is the same game as Game 4.

Game $6.j$ **(Switch constrained proof witnesses for** $j \in [0, q_1(\lambda)]$**):** For each constrain query $\jmath < j$ with constraint $C_\jmath$, we use the witness $\mathsf{w}_{C_\jmath} := (1, 2, k_{x^*}, k_{x^*}, r_1^0, r_2^0, k^1, r_1^1, r_2^1)$ instead of $\mathsf{w}_{C_\jmath} := (1, 3, k_{x^*}, k^0, r_1^0, r_3^0, k^1, r_1^1, r_3^1)$ where $k^1 := \mathsf{CPRF.Constrain}(k^0, C_\jmath)$. Note that both witnesses are valid for the statement $\mathsf{stmt}_{C_\jmath, c_1^0, c_2^0, c_3^0, c_1^1, c_2^1, c_3^1}$. Again, this follows from the puncture invariance of CPRF. Consequently, the distinguishing advantage of the adversary between this game and the previous is at most

$$|\Pr[\text{Game } 6.j] - \Pr[\text{Game } 6.(j-1)]| \leq \epsilon_{\mathsf{NIWI}}(\lambda) \tag{16}$$

by the witness indistinguishability of NIWI. Overall, we find

$$|\Pr[\text{Game } 6.0] - \Pr[\text{Game } 6.q_1(\lambda)]| \leq q_1(\lambda) \cdot \epsilon_{\mathsf{NIWI}}(\lambda) \ . \tag{17}$$

Moreover, Game 6.0 is the same game as Game $5.q_2(\lambda)$.

Game 7 **(Switch the third commitment to a punctured key):** In this game, the third commitment is generated as $c_3^0 := \mathsf{COM.Com}(k_{x^*}; r_3^0)$ where $k_{x^*} := \mathsf{CPRF.Constrain}(k, \{x^*\})$ instead of $c_3^0 := \mathsf{COM.Com}(k^0; r_3^0)$. The distinguishing advantage of the adversary between this game and the previous is at most

$$|\Pr[\text{Game } 7] - \Pr[\text{Game } 6.q_1(\lambda)]| \leq \epsilon_{\mathsf{COM}}(\lambda) \tag{18}$$

because the randomness r_3^0 is not used in the NIWI proofs.

Game 8 **(Switch the challenge image to random):** In this game, we sample the challenge image $y^* \leftarrow \mathcal{Y}$ instead of setting it as the real evaluation $(y^*, \pi^*) = \mathsf{CVRF.Eval}(\mathsf{sk}, x^*)$. Note that the previous hybrid game does not use the actual CPRF key k^0 anymore, but only the punctured key k_{x^*}. Hence, the distinguishing advantage of the adversary between this game and the previous is at most

$$|\Pr[\text{Game } 8] - \Pr[\text{Game } 7]| \leq \epsilon_{\mathsf{CPRF}}(\lambda) \ . \tag{19}$$

Overall, we find that the distinguishing advantage of the adversary in the original game and the last hybrid game is at most

$$\begin{aligned}
&|\Pr[\text{Game } 0] - \Pr[\text{Game } 8]| && (20)\\
&\leq 3 \cdot \epsilon_{\mathsf{COM}}(\lambda) + 2 \cdot q_1(\lambda) \cdot \epsilon_{\mathsf{NIWI}}(\lambda) + 2 \cdot q_2(\lambda) \cdot \epsilon_{\mathsf{NIWI}}(\lambda) + \epsilon_{\mathsf{CPRF}}(\lambda) && (21)\\
&\leq 3 \cdot \epsilon_{\mathsf{COM}}(\lambda) + 2 \cdot q(\lambda) \cdot \epsilon_{\mathsf{NIWI}}(\lambda) + \epsilon_{\mathsf{CPRF}}(\lambda) \ . && (22)
\end{aligned}$$

By reverting all previous game hops, we can get back the original pseudorandomness game (with challenge bit $b = 1$). Hence, we need to double the terms $3 \cdot \epsilon_{\mathsf{COM}}(\lambda) + 2 \cdot q(\lambda) \cdot \epsilon_{\mathsf{NIWI}}(\lambda)$ to bound the final advantage of the adversary. □

Theorem 8 (Unique Provability)**.** *Let* COM *be a perfectly binding non-interactive commitment scheme. Let* NIWI *be a perfectly sound NIWI. Let* CPRF *be a* 1*-delegable CPRF. Then the CVRF* CVRF *in Construction 2 is unqiuely provable.*

Proof. Let vk be any (possibly malformed) verification key. Let x be an arbitrary preimage, $y \neq y'$ images and $\pi = (r_2, r_3, (C^i, c_1^i, c_2^i, c_3^i, \pi^i)_{i\in[j]}), \pi' = (r_2', r_3', (C'^i, c_1'^i, c_2'^i, c_3'^i, \pi'^i)_{i\in[j]})$ be any (possibly malformed) proofs. The perfect binding property of COM fixes k_1^0, k_2^0, k_3^0 that are committed to in vk. Suppose for contradiction that

$$\mathsf{CVRF.Verify}(\mathsf{vk}, x, y, \pi) = \mathsf{ACCEPT} \wedge \mathsf{CVRF.Verify}(\mathsf{vk}, x, y', \pi') = \mathsf{ACCEPT} \ . \tag{23}$$

This implies that $x \notin C^j$ and $x \notin C'^j$. Moreover, y is the first output of $\mathsf{CPRF.Eval}(k_i^0, x)$ and $\mathsf{CPRF.Eval}(k_j^0, x)$ for some $i \neq j$ and y' is the first output of $\mathsf{CPRF.Eval}(k_{i'}^0, x)$ and $\mathsf{CPRF.Eval}(k_{j'}^0, x)$ for some $i' \neq j'$. Hence, there exists some $\kappa \in \{1, 2, 3\}$ such that y and y' are both the first output of $\mathsf{CPRF.Eval}(k_\kappa^0, x)$ which contradicts $y \neq y'$. □

Theorem 9 (Delegability)**.** *Let* NIWI *be a perfectly complete NIWI. Let* CPRF *be a d-delegable CPRF. Then the CVRF* CVRF *in Construction 2 is d-delegable.*

Proof. For any (honestly generated) $(\mathsf{pp}, \mathsf{sk}_\emptyset, \mathsf{vk}, \pi_\emptyset) \leftarrow \mathsf{CVRF.Setup}(1^\lambda)$, any constraints $C_0 \subseteq \cdots \subseteq C_d$, any preimage $x \in \mathcal{X} \setminus C_d$ (outside all constraints), any (honestly generated) constrained keys $(\mathsf{sk}_{C_i}, \pi_{C_i}) \leftarrow \mathsf{CVRF.Constrain}(\mathsf{sk}_{C_{i-1}}, C_i)$, any (honestly generated) images and proofs $(y^i, \pi^i) \leftarrow \mathsf{CVRF.Eval}(\mathsf{sk}_{C_{i-1}}, x)$, it follows from the perfect completeness that all NIWI proofs verify. Moreover, the condition $C_0 \subseteq \cdots \subseteq C_d$ in line 6 of $\mathsf{VerifyC}$ and line 5 of Verify is fulfilled by definition. Hence, all checks in both Verify and $\mathsf{VerifyC}$ pass. □

Theorem 10 (Functionality Binding)**.** *Let* COM *be a perfectly binding non-interactive commitment scheme. Let* NIWI *be a perfectly sound NIWI. Then the CVRF* CVRF *in Construction 2 is functionality binding.*

Proof. For contradiction suppose there exist public parameters pp, verification key vk, secret keys $\mathsf{sk}_C = k, \mathsf{sk}_C' = k'$, preimage x, image-proof-pairs $(y, \pi) \leftarrow \mathsf{CPRF.Eval}(\mathsf{sk}_C, x)$ and $(y', \pi') \leftarrow \mathsf{CPRF.Eval}(\mathsf{sk}_C', x)$ such that

$$\begin{aligned} &\mathsf{CVRF.VerifyC}_{\mathsf{pp}}(\mathsf{vk}, C, \mathsf{sk}_C, \pi_C) = \mathsf{CVRF.VerifyC}_{\mathsf{pp}}(\mathsf{vk}, C, \mathsf{sk}_C', \pi_C') = \mathsf{ACCEPT} \\ &\text{and} \quad y \neq y' \ . \end{aligned}$$

Since $y = \mathsf{CPRF.Eval}(\mathsf{sk}_C, x)$ and $\mathsf{CPRF.Eval}$ is deterministic, it follows that $\mathsf{sk}_C \neq \mathsf{sk}_C'$. Let k_1^0, k_2^0, k_3^0 be the keys that are contained in the verfication key (by perfect binding of COM). Because $\mathsf{CVRF.VerifyC}_{\mathsf{pp}}(\mathsf{vk}, C, \mathsf{sk}, \pi_C) = 1$ and $\mathsf{CVRF.VerifyC}_{\mathsf{pp}}(\mathsf{vk}, C, \mathsf{sk}', \pi') = 1$ the perfect soundness of NIWI, it follows that $k = \mathsf{CPRF.Constrain}(k_i^0, C) = \mathsf{CPRF.Constrain}(k_j^0, C)$ for some $i \neq j$ and $k' = \mathsf{CPRF.Constrain}(k_{i'}^0, C) = \mathsf{CPRF.Constrain}(k_{j'}^0, C)$ for some $i' \neq j'$. Consequently, there exists some $\kappa \in \{1, 2, 3\}$ such that $\mathsf{sk}_C = k = \mathsf{CPRF.Constrain}(k_\kappa^0, C) = k' = \mathsf{sk}_C'$ which contradicts $\mathsf{sk}_C \neq \mathsf{sk}_C'$. □

Remark 3 (Limits of the NIWI-Based CVRF)**.** *The NIWI-based construction does not fulfill constraint-hiding in the sense of Def. 6. This is because the proofs generated by* $\mathsf{Constrain}$ *or* Eval *leak the full history of constraints. If a CVRF is not constraint-hiding, then the selective security game as stated in Def. 5 does not capture all adversarial behavior (as discussed in more detail in Remark 1).*

In fact, the NIWI-based CVRF as stated in Construction 2 cannot *support more elaborate adversarial queries. The issue is that the construction* overcommits *to the secret*

key when given, e.g., the elaborate, yet valid query "Constrain with $\emptyset$, do not output the constrained key, but use the key to evaluate at some point $x \neq x^*$ and give me the output." Constraining with $\emptyset$ results in another commitment to the secret key by which the above proof strategy breaks down, and we cannot replace this key by a punctured one.

One can conjecture ways to circumvent this issue (e.g., using NIWI proofs of past constraints as witnesses to later NIWI proofs). However, due to the perfect soundness of the NIWI, it seems that the proof size (at least when adapting the blueprint of [GHKW17, Bit20] to produce new commitments for constrained keys) must unavoidably leak the number of prior constraints, thereby violating constraint-hiding. Thus, one cannot directly appeal to constraint-hiding and would have to prove that such leakage is not an issue.

The main takeaway is that NIWI-based CVRFs tolerating elaborate queries seemingly require non-trivial effort and do not follow immediately from [GHKW17, Bit20]. We leave it as an open question to construct such a CVRF using NIWIs.

D Generalizing CVRF With Respect to Any Algebraic PRG

We explain here how to generalize our CVRF construction for any family of quadratic PRG functions. This may be practical if one wants to use PRGs that can be evaluated very fast, or if one wants to use a special PRG candidate with (hypothetically) higher security than uniformly random quadratic polynomials.

As far as it concerns the construction of CVRF, it suffices to replace the uniformly random polynomials $(G_0^{(i)}, G_1^{(i)})$ sampled by our CVRF in § 3 in its setup algorithm with polynomials of special distributions, without changing any other of its algorithms. When doing so, one only has to adapt the recursive DDH assumption to the corresponding family of PRGs. To prove the soundness of recursive DDH assumptions that are with respect to special distributions of $(G_0, G_1) \in Q_{p,n}^2$, we need to distill essential properties a PRG candidate needs to have to guarantee security in the generic group model. We call these properties *Pseudorandomness*, *Separability* and *Resistance* and list them in App. D.1.

We give the modified CVRF construction with respect to any candidate of quadratic PRG functions in App. D.2, and derive a modified recursive DDH assumption with respect to special PRGs, from which the security of the modified CVRF can be proven. The security proof of the modified CVRF is then identical to the proof of Thm. 4.

In App. D.3, we prove the correctness of the modified recursive DDH assumption in the generic group model assuming that the underlying PRG family is pseudorandom, resistant and separable.

D.1 Resistant and Separable PRGs

Definition 14. Let $\mathcal{Q}$ be an (efficiently computable) distribution over $Q_{p,n}^2$. I.e., each $(G_0, G_1) \leftarrow \mathcal{Q}$ is a pair of two maps $G_0, G_1 : \mathbb{Z}_p^n \to \mathbb{Z}_p^n$ whose outputs are computed by degree-2 polynomials over $\mathbb{Z}_p$.

1. We call $\mathcal{Q}$ *symmetric* if we have for all $G_0, G_1 \in Q_{p,n}^2$

$$\Pr[(G_0, G_1) \leftarrow \mathcal{Q}] = \Pr[(G_1, G_0) \leftarrow \mathcal{Q}].$$

2. For $0 \leq \eta \leq 1$, we call $\mathcal{Q}$ *η-separable* if we have

$$\Pr[\exists \boldsymbol{s}, \boldsymbol{w} \in \mathbb{Z}_p^n, \exists f \in \mathrm{span}_{\mathbb{Z}_p}[G_0(X) - \boldsymbol{s}, G_1(X) - \boldsymbol{w}] \mid f \neq 0, \deg f \leq 1] \leq 1 - \eta.$$

(Note that the above requirement is equivalent to

$$\Pr[\dim_{\mathbb{Z}_p} \left(\mathrm{span}_{\mathbb{Z}_p}[g_1, \ldots, g_{2n}, X_1, \ldots, X_n, 1]\right) = 3n + 1] \geq \eta$$

where $g_1, \ldots, g_{2n}$ are polynomials computing (G_0, G_1).)

3. For $\mu \geq 0$ and $t > 0$, we call $\mathcal{Q}$ *(t, μ)-pseudorandom* if we have for each adversary $\mathcal{A}$ running in time t

$$|\Pr[\mathcal{A}((G_0, G_1), (G_0(\boldsymbol{x}), G_1(\boldsymbol{x}))) = 1] - \Pr[\mathcal{A}((G_0, G_1), \boldsymbol{y}) = 1]| \leq \mu$$

where $(G_0, G_1) \leftarrow \mathcal{Q}$, $\boldsymbol{x} \leftarrow\!\!\$\ \mathbb{Z}_p^n$ and $\boldsymbol{y} \leftarrow\!\!\$\ \mathbb{Z}_p^{2n}$.

4. For $\mu \geq 0$ and $t > 0$, we call $\mathcal{Q}$ *(t, μ)-resistant* if we have for each adversary $\mathcal{A}$ running in time t

$$\Pr\begin{bmatrix} h \in \mathbb{Z}_p[X_1, \ldots, X_n]^{\leq 2}, h(\boldsymbol{x}) = 0, \\ h \notin \mathrm{span}_{\mathbb{Z}_p}[G_0(X) - G_0(\boldsymbol{x}), G_1(X) - G_1(\boldsymbol{x})] \end{bmatrix} - \frac{2}{p} \leq \mu$$

where $(G_0, G_1) \leftarrow \mathcal{Q}$, $\boldsymbol{x} \leftarrow\!\!\$\ \mathbb{Z}_p^n$ and $h \leftarrow \mathcal{A}((G_0, G_1), (G_0(\boldsymbol{x}), G_1(\boldsymbol{x})))$.

In § 5, we studied the case where $\mathcal{Q}$ is the uniform distribution over $Q_{p,n}^2$. In this case, $\mathcal{Q}$ is naturally symmetric, and we showed that it is $(1 - 2n/p)$-separable. Pseudorandomness of the uniform distribution over $Q_{p,n}^2$ is equivalent to the MQ assumption, and resistance of it is equivalent to the $\mathsf{MQ}+$ assumption.

D.2 Construction for General PRGs

As in § 3, we denote for $x = x_1 \cdots x_{\ell'} \in \{0,1\}^{\ell'}$, $\ell' \leq \ell$, and $\mathcal{G} = ((G_0^{(1)}, G_1^{(1)}), \ldots, (G_0^{(\ell)}, G_1^{(\ell)})) \leftarrow \mathcal{Q}^\ell$

$$G_x := G_{x_{\ell'}}^{(\ell')} \circ \cdots \circ G_{x_1}^{(1)}.$$

The new CVRF construction is almost identical with the one given in § 3, with the minor difference that Setup now samples $\mathcal{G}$ from $\mathcal{Q}^\ell$ instead of $Q_{p,n}^{\ell \times 2}$. We detail this in Fig. 10.

To prove the security of the modified CVRF CVRF, we need to modify the underlying recursive DDH assumption as well.

Assumption 4 (Recursive DDH for General PRGs)**.** Fix $n, m_1, m_2 \in \mathrm{poly}(\lambda)$, a prime $p \leq 2^{\mathrm{poly}(\lambda)}$ and a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ of groups of order p with generators $\mathsf{g}_1 \in \mathbb{G}_1$, $\mathsf{g}_2 \in \mathbb{G}_2$.

Fix a distribution $\mathcal{Q}$ over $Q_{p,n}^2$ that is symmetric, separable, pseudorandom and resistant in the sense of Def. 14.

Let Samp be an algorithm that on input $1^\lambda, p, n$, outputs descriptions of

$\mathsf{CVRF.Setup}(1^\lambda)$:
01 Sample $\boldsymbol{s} \leftarrow_\$ \mathbb{Z}_p^n$; Set $\mathsf{sk}_\epsilon = \boldsymbol{s}$ 02 Set $\mathsf{vk} = \mathsf{g}_2^{\boldsymbol{s}}$ 03 Sample degree-2 maps $\mathcal{G} \leftarrow_\$ \mathcal{Q}^\ell$; Set $\mathsf{pp} = \mathcal{G}$ 04 Output $(\mathsf{pp}, \mathsf{sk}_\epsilon, \mathsf{vk}, \pi_\epsilon = \emptyset)$

Figure 10: Pseudocode of the new Setup algorithm for CVRF with general PRG distribution $\mathcal{Q}$. The only difference to the CVRF construction in Fig. 2 has been put in a gray box.

1. an efficiently computable function $c : \mathbb{Z}_p^n \to \mathbb{Z}_p^n$,
2. an efficiently computable function $h_0 : \mathbb{Z}_p^n \to \mathbb{Z}_p^{m_0}$,
3. an efficiently computable function $h_1 : \mathbb{Z}_p^n \to \mathbb{Z}_p^{m_1}$,
4. and an efficiently computable function $h_2 : \mathbb{Z}_p^n \to \mathbb{Z}_p^{m_2}$.

For an adversary $\mathcal{A}_{\mathsf{if}}$, we define its advantage in the if-challenge by

$$\begin{aligned}\mathrm{Adv}_{\mathsf{if}}^{\mathcal{A}_{\mathsf{if}}} = |&\Pr[\mathcal{A}_{\mathsf{if}}(c, h_0, h_1, h_2, h_0(\boldsymbol{s}), \mathsf{g}_1^{h_1(\boldsymbol{s})}, \mathsf{g}_2^{h_2(\boldsymbol{s})}, \mathsf{g}_2^{\boldsymbol{s}}, \mathsf{g}_2^{c(\boldsymbol{s})}) = 1] \\ &- \Pr[\mathcal{A}_{\mathsf{if}}(c, h_0, h_1, h_2, h_0(\boldsymbol{s}), \mathsf{g}_1^{h_1(\boldsymbol{s})}, \mathsf{g}_2^{h_2(\boldsymbol{s})}, \mathsf{g}_2^{\boldsymbol{s}}, \mathsf{g}_2^{\boldsymbol{r}}) = 1]|\end{aligned}$$

where the probabilities are taken over $(c, h_0, h_1, h_2) \leftarrow \mathsf{Samp}(1^\lambda, p, n)$, $\boldsymbol{s} \leftarrow_\$ \mathbb{Z}_p^n$ and $\boldsymbol{r} \leftarrow_\$ \mathbb{Z}_p^n$. For an adversary $\mathcal{A}_{\mathsf{then}}$, we define its advantage in the then-challenge by

$$\begin{aligned}\mathrm{Adv}_{\mathsf{then}}^{\mathcal{A}_{\mathsf{then}}} = |&\Pr[\mathcal{A}_{\mathsf{then}}(c, h_0, h_1, h_2, G_0, G_1, h_0(\boldsymbol{s}), \mathsf{g}_1^{h_1(\boldsymbol{s})}, \mathsf{g}_2^{h_2(\boldsymbol{s})}, \mathsf{g}_2^{\boldsymbol{s}}, \mathsf{g}_1^{\boldsymbol{x}}, G_1(\boldsymbol{x}), \mathsf{g}_2^{c(\boldsymbol{s})}) = 1] \\ &- \Pr[\mathcal{A}_{\mathsf{then}}(c, h_0, h_1, h_2, G_0, G_1, h_0(\boldsymbol{s}), \mathsf{g}_1^{h_1(\boldsymbol{s})}, \mathsf{g}_2^{h_2(\boldsymbol{s})}, \mathsf{g}_2^{\boldsymbol{s}}, \mathsf{g}_1^{\boldsymbol{x}}, G_1(\boldsymbol{x}), \mathsf{g}_2^{\boldsymbol{r}}) = 1]|\end{aligned}$$

where $\boldsymbol{s} = G_0(\boldsymbol{x})$, and the probabilities are taken over $(c, h_0, h_1, h_2) \leftarrow \mathsf{Samp}(1^\lambda, p, n)$, $\boldsymbol{x} \leftarrow_\$ \mathbb{Z}_p^n$, $(G_0, G_1) \leftarrow \mathcal{Q}$ and $\boldsymbol{r} \leftarrow_\$ \mathbb{Z}_p^n$.

Finally, fix non-negative functions $t = t(\lambda)$, $q = q(\lambda)$, $\mu = \mu(\lambda)$. The *recursive Decisional Diffie-Hellman* assumption *with respect to the PRG family* $\mathcal{Q}$, denoted by (t, q, μ)-$\mathsf{rDDH}_{\mathcal{Q}}$, states that for all samplers of the above form that run in time t and for all then-adversaries $\mathcal{A}_{\mathsf{then}}$, there exists an if-adversary $\mathcal{A}_{\mathsf{if}}$ s.t.

$$\mathrm{Adv}_{\mathsf{if}}^{\mathcal{A}_{\mathsf{if}}}(\lambda) \geq \mathrm{Adv}_{\mathsf{then}}^{\mathcal{A}_{\mathsf{then}}}(\lambda) - \mu(\lambda) \qquad \text{and} \qquad \mathrm{time}(\mathcal{A}_{\mathsf{if}}) \leq \mathrm{time}(\mathcal{A}_{\mathsf{then}}) + q.$$

We will show in App. D.3 that $(t'', t'' + O(qn^6), q\mu_{\mathsf{res}} + \mu_{\mathsf{prg}} + \mu_{\mathsf{sep}} + q/p)$-$\mathsf{rDDH}_{\mathcal{Q}}$ holds in the GGM if $\mathcal{Q}$ is $(1 - \mu_{\mathsf{sep}})$-separable, $(t + t'' + O(qn^6), \mu_{\mathsf{prg}})$-pseudorandom and $(t + t'' + O(qn^6), \mu_{\mathsf{res}})$-resistant (and if the number of group queries and bit operations of adversaries is bounded by q and t, respectively).

Note that delegability, unique provability, functionality binding and constraint-hiding for CVRF with the modified setup algorithm follows from Thm. 3, as those properties are independent of the underlying PRG.

The proof of security of CVRF is identical to the corresponding proofs of our construction based on uniformly random polynomials in § 3. The only difference appears in replacing the distribution of $(G_0^{(i)}, G_1^{(i)})$. Note that we require that $\mathcal{Q}$ is symmetric. This allows us

to always assume—without loss of generality—that the adversary challenges us on the point $x^* = 0^\ell$.

Hence, we can recollect:

Corollary 1. *1.* CVRF *is a* Prefix-Constrained Verifiable Function *as in Def. 5, i.e., it satisfies* Delegability, Functionality Binding *and* Unique Provability. *Additionally, it is* Constraint-Hiding.

2. Under $(t+\ell q+\ell n t_{\mathsf{pair}}, \mu)$-$\mathsf{DSDH}$ *and* $(O(\ell t_{\mathcal{Q}}), q, \mu')$-$\mathsf{rDDH}_{\mathcal{Q}}$ *the function* $\mathsf{CVRF}\colon \{0,1\}^\ell \to \mathbb{G}_2^n$ *is a* $(t, \mu + \mu'\ell)$*-CVRF, where* $t_{\mathcal{Q}}$ *and* t_{pair} *are the time complexities to sample from* $\mathcal{Q}$ *and to evaluate* $e : \mathbb{G}_1^2 \to \mathbb{G}_2$ *once, respectively.*

D.3 Proving $\mathsf{rDDH}_{\mathcal{Q}}$ in the Generic Group Model

The following theorem is analogous to Thm. 5:

Theorem 11. *Let* $t, t', t'' > 0$*. Let* Samp *be a sampling algorithm with time complexity* t'' *for Ass. 4 that outputs functions* h_0, h_1, h_2, c *that can be evaluated in time* t''*.*

Let $\mathcal{Q}$ *be a distribution over* $Q_{p,n}^2$ *that is symmetric,* $(1-\mu_{\mathsf{sep}})$*-separable,* (t', μ_{prg})*-pseudorandom and* (t', μ_{res})*-resistant for* $t' = t + t'' + O(qn^6)$*.*

Let $\mathcal{A}_{\mathsf{then}}$ *be a generic adversary against the* then*-challenge of Ass. 4 with advantage* $\mu + q\mu_{\mathsf{res}} + \mu_{\mathsf{prg}} + \mu_{\mathsf{sep}} + 3q/p$ *and time complexity* t *that makes* q *group-queries.*

Then, there exists an adversary $\mathcal{A}_{\mathsf{if}}$ *on the* if*-challenge of Ass. 4 that makes* q *group-queries and has an advantage of* μ *and a time complexity of* t'*.*

Proof. To prove Thm. 11, we consider the hybrids given in Fig. 11. Note that the hybrids of Fig. 11 are identical to the hybrids of Fig. 4, used in the proof of Thm. 5, with the only exception how (G_0, G_1) is sampled by the game setup. Consequently, the proof of Thm. 11 is analogous to the proof of Thm. 5. We only need to address in which game hops we use which property of $\mathcal{Q}$:

1. By the resistance of $\mathcal{Q}$, we have

$$|\Pr[\mathsf{G}_{i-1}^{\mathcal{A}_{\mathsf{then}}}(1^\lambda) = 1] - \Pr[\mathsf{G}_i^{\mathcal{A}_{\mathsf{then}}}(1^\lambda) = 1]| \leq \mu_{\mathsf{res}} + \frac{2}{p}$$

   for $i = 1, \ldots, q$. Indeed, G_{i-1} and G_i only differ in how they handle the i-th group query of $\mathcal{A}_{\mathsf{then}}$. If there would exist some noticeable difference, one could proceed as in the proof of Lem. 1 and reduce the resistance-game of $\mathcal{Q}$ to distinguishing $\mathsf{G}_{i-1}^{\mathcal{A}_{\mathsf{then}}}$ and $\mathsf{G}_i^{\mathcal{A}_{\mathsf{then}}}$.

2. By the pseudorandomness of $\mathcal{Q}$, we have

$$|\Pr[\mathsf{G}_q^{\mathcal{A}_{\mathsf{then}}}(1^\lambda) = 1] - \Pr[\mathsf{G}_{q+1}^{\mathcal{A}_{\mathsf{then}}}(1^\lambda) = 1]| \leq \mu_{\mathsf{prg}}.$$

   Indeed, G_q and G_{q+1} only differ in how they generate $\boldsymbol{s}$ and $\boldsymbol{w}$. Analogous to the proof of Lem. 2, one can reduce the pseudorandomness game of $\mathcal{Q}$ to the problem of distinguishing $\mathsf{G}_q^{\mathcal{A}_{\mathsf{then}}}$ and $\mathsf{G}_{q+1}^{\mathcal{A}_{\mathsf{then}}}$.

Game $\mathsf{G}_i^{\mathcal{A}_{\mathsf{then}}}(1^\lambda)$, $i \in \{0,\dots,q+2\}$, **with adversary** $\mathcal{A}_{\mathsf{then}}$**:**

Initialize generic groups $\mathbb{G}_1 = \langle \mathsf{g}_1 \rangle$, $\mathbb{G}_2 = \langle \mathsf{g}_2 \rangle$
Set $j = 0$
Sample $b \leftarrow\!\!\$\ \{0,1\}$
Sample $(c, h_0, h_1, h_2) \leftarrow \mathsf{Samp}(1^\lambda, p, n)$
Sample $\boldsymbol{x} \leftarrow\!\!\$\ \mathbb{Z}_p^n$, $(G_0, G_1) \leftarrow\!\!\$\ \mathcal{Q}$
If $i \le q$, set $\boldsymbol{s} = G_0(\boldsymbol{x})$ and $\boldsymbol{w} = G_1(\boldsymbol{x})$
If $i \ge q+1$, sample $\boldsymbol{s}, \boldsymbol{w} \leftarrow\!\!\$\ \mathbb{Z}_p^n$
If $b = 1$, sample $\boldsymbol{c} \leftarrow\!\!\$\ \mathbb{Z}_p^n$; else, set $\boldsymbol{c} = c(\boldsymbol{s})$
Run $b' \leftarrow \mathcal{A}_{\mathsf{then}}^{\mathsf{R}}(c, h_0, h_1, h_2, G_0, G_1, h_0(\boldsymbol{s}), \mathsf{g}_1^{H^{(1)}}, \mathsf{g}_2^{H^{(2)}}, \mathsf{g}_2^S, \mathsf{g}_1^X, \boldsymbol{w}, \mathsf{g}_2^C)$
Return 1, if $b = b'$; else, return 0

Group Query Subroutine $\mathsf{R}(f \in \mathbb{Z}_p[H^{(1)}, H^{(2)}, X, S, C])$**:**

Increase $j = j + 1$
If f is not admissible or $j > q$, return $\perp$
If $j > i \le q+1$, do the following:
    If $f(h_1(\boldsymbol{s}), h_2(\boldsymbol{s}), \boldsymbol{x}, \boldsymbol{s}, \boldsymbol{c}) = 0$, return 1; else, return 0;
If $j \le i \le q+1$, do the following:
    If $f(h_1(\boldsymbol{s}), h_2(\boldsymbol{s}), X, \boldsymbol{s}, \boldsymbol{c}) \in \mathrm{span}_{\mathbb{Z}_p}[G_0(X) - \boldsymbol{s}, G_1(X) - \boldsymbol{w}]$, return 1
    Else, return 0
If $i = q+2$, do the following:
    Compute $f' \in B$ s.t. $f - f' \in \mathrm{span}_{\mathbb{Z}_p}[G_0(X) - S, G_1(X) - \boldsymbol{w}]$
    If such an f' does not exist, return 0
    Draw $\boldsymbol{x}' \leftarrow\!\!\$\ \mathbb{Z}_p^n$
    If $f'(h_1(\boldsymbol{s}), h_2(\boldsymbol{s}), \boldsymbol{x}', \boldsymbol{s}, \boldsymbol{c}) = 0$, return 1; else, return 0

Figure 11: Hybrid Games for the proof of Thm. 11. The only difference to the hybrids of Fig. 4 is marked in gray.

3. By the separability of $\mathcal{Q}$, we have

$$|\Pr[\mathsf{G}_{q+1}^{\mathcal{A}_{\mathsf{then}}}(1^\lambda) = 1] - \Pr[\mathsf{G}_{q+2}^{\mathcal{A}_{\mathsf{then}}}(1^\lambda) = 1]| \le \mu_{\mathsf{sep}} + q/p.$$

   For this, note that we have with probability $(1 - \mu_{\mathsf{sep}})$ that there are no $\boldsymbol{s}, \boldsymbol{w} \in \mathbb{Z}_p^n$ and $f \in \mathrm{span}_{\mathbb{Z}_p}[G_0(X) - \boldsymbol{s}, G_1(X) - \boldsymbol{w}]$ s.t. f is non-zero and of degree ≤ 1. From this point on, the proof of Lem. 4 is independent of the distribution of (G_0, G_1) and, hence, can also be applied here.

It follows that $\mathcal{A}_{\mathsf{then}}$'s advantage at winning G_{q+2} is at least μ. From here on, the distribution of (G_0, G_1) is again not relevant, and we can use the same reduction $\mathcal{R}$ as in the proof of Thm. 5 (with the only difference that $\mathcal{R}$ samples (G_0, G_1) from $\mathcal{Q}$). □