An extended abstract of this work appears in the *9th Conference on Security and Cryptography for Networks (SCN'14)*, Michel Abdalla, Roberto De Prisco (Eds.), Springer Verlag, LNCS 8642. This is the full version.

Constrained Verifiable Random Functions

Georg Fuchsbauer*

Institute of Science and Technology Austria
georg.fuchsbauer@ist.ac.at

Abstract

We extend the notion of verifiable random functions (VRF) to *constrained VRFs*, which generalize the concept of constrained pseudorandom functions, put forward by Boneh and Waters (Asiacrypt'13), and independently by Kiayias et al. (CCS'13) and Boyle et al. (PKC'14), who call them delegatable PRFs and functional PRFs, respectively. In a standard VRF the secret key sk allows one to evaluate a pseudorandom function at any point of its domain; in addition, it enables computation of a non-interactive proof that the function value was computed correctly. In a constrained VRF from the key sk one can derive constrained keys sk_S for subsets S of the domain, which allow computation of function values and proofs only at points in S.

After formally defining constrained VRFs, we derive instantiations from the multilinear-maps-based constrained PRFs by Boneh and Waters, yielding a VRF with constrained keys for any set that can be decided by a polynomial-size circuit. Our VRFs have the same function values as the Boneh-Waters PRFs and are proved secure under the same hardness assumption, showing that verifiability comes at no cost. Constrained (functional) VRFs were stated as an open problem by Boyle et al.

1 Introduction

Verifiable random functions. A pseudorandom function (PRF) [GGM86] is an efficiently computable keyed function $F\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ for which, when the seed k is chosen at random, no efficient attacker should be able to distinguish $F(k,x)$ from a random value, even when given oracle access to $F(k,\cdot)$ at any other point. This fundamental primitive in cryptography was extended to verifiable random functions (VRF) by Micali, Rabin and Vadhan [MRV99]. In a VRF a secret key sk, which is set up together with a public key pk, allows evaluation of F and furthermore computation of a non-interactive proof that the computed value y matches $F(sk,x)$. Verification of the proof must be done with respect to the public key pk only; in particular, we cannot make use of a common reference string (CRS). The proofs should remain sound even when pk was computed maliciously and $F(sk,x)$ should remain pseudorandom even when an adversary can query values of F and proofs for them at any other point.

The first VRF schemes were based on bilinear maps, such as [Lys02, Dod03, DY05]. Efficient schemes have proved difficult to construct, in particular ones with large domains based on non-interactive assumptions, and were only proposed from 2010 on [HW10, BMR10, ACF13]. VRFs have turned out to be a useful building block, e.g. in the construction of zero-knowledge proofs and databases [MR01, Lis05] and electronic payment schemes [MR02, BCKL09], to name a few.

*Supported by the European Research Council, ERC Starting Grant (259668-PSPC).

Constrained VRFs. Boneh and Waters [BW13] define a new notion of PRFs, which they call constrained PRFs and which was concurrently introduced as delegatable PRFs by Kiayias, Papadopoulos, Triandopoulos and Zacharias [KPTZ13], and as functional PRFs by Boyle, Goldwasser and Ivan [BGI14]. While a key k for a PRF enables evaluation of the function F at all points of its domain $\mathcal{X}$, a constrained PRF allows one to derive constrained keys from k. A constrained key k_S corresponds to a set $S \subseteq \mathcal{X}$ and allows computation of $F(k, x)$ only for $x \in S$.

Pseudorandomness requires that given an oracle for function values at points of the adversary's choice and an oracle for constrained keys for sets of its choice, values of $F(k, \cdot)$ at points outside the queried sets and different from the queried points should still be indistinguishable from random. That is, after querying keys for $S_1, \ldots, S_q$ and functions values at $x_1, \ldots, x_p$, the value $F(k, x)$ should be indistinguishable from random for all $x \notin \bigcup_{i=1}^{q} S_i \cup \{x_1, \ldots, x_p\}$. Constrained PRFs were used to construct broadcast encryption and identity-based non-interactive key exchange in [BW13]. In particular *punctured PRFs* have proved to be a powerful tool in combination with indistinguishability obfuscation [GGH+13b], leading to solutions of longstanding open problems, such as deniable encryption [CDNO97] in [SW13] and instantiating full-domain hash [BR93] in [HSW13b].

We unify VRFs and constrained PRFs by adding the possibility to derive constrained keys to the notion of VRFs. We then construct constrained VRF schemes based on the Boneh-Waters constrained PRFs which are defined using multilinear maps [BS02, GGH13a, CLT13]. Our second scheme allows derivation of constrained keys for any subset of the domain that can be decided by a boolean circuit of polynomial size.

Verifiable random functions turned out a lot harder to construct than PRFs. While the Dodis-Yampolskiy VRF [DY05] only supports domains of polynomial size, it requires a q-type assumption (where the parameter q of the assumption upper-bounds how many queries an adversary can make). Hohenberger and Waters [HW10] proposed the first VRF for large domains, whose function values are defined analogously to those of the PRF by Naor and Reingold [NR97], but lifted to the target group of a bilinear map. These maps are then used to verify the proofs of correct function evaluation. Hohenberger and Waters prove their construction secure under a non-standard q-type assumption, while the Naor-Reingold PRF is proved secure under the decisional Diffie-Hellman (DDH) assumption.

Using multilinear maps, the situation is different: Our VRF constructions support large input spaces when using complexity leveraging (see below), and we prove their security under the same assumption on which pseudorandomness of the Boneh-Waters constrained PRFs rely: the DDH assumption adapted to the multilinear-map environment. We moreover show that we do not need to lift the function values "up one level": our VRF values are defined exactly as the Boneh-Waters PRF values. We thus show how to add verifiability to the constrained PRFs from [BW13] without changing the PRF itself, nor using a different assumption to prove pseudorandomness.

Our contribution. We first formalize the notion of constrained VRFs by extending the model for standard VRFs. In addition to Setup, Prove and Verify, we define an algorithm Constrain, which allows to derive constrained keys. We adapt the security notions of *provability*, *uniqueness* and *pseudorandomness* to the constrained setting and define a new security notion. It requires that a proof produced by a constrained key should be distributed like proofs computed using the actual secret key. A constrained key sk_S behaves thus exactly like the key sk on the subset S of the domain.

We present two multilinear-maps-based instantiations of constrained VRFs with input space $\mathcal{X} := \{0,1\}^n$ for different systems of sets for which constrained keys can be derived:

- Bit-fixing VRF: Constrained keys can be derived for any set $S_{\mathbf{v}} \subseteq \{0,1\}^n$, described by a vector

$\mathbf{v} \in \{0, 1, ?\}$ as the set of all strings which match $\mathbf{v}$ at all coordinates that are not '?'.

- Circuit-constrained VRF: In our second construction keys can be derived for any set that is decidable by a polynomial-size circuit C. More precisely, a key sk_C, derived from sk for a circuit C, enables computation of $F(sk, x)$ and a proof for all x for which $C(x) = 1$.

Both our schemes are directly derived from the constructions of constrained PRFs given by Boneh and Waters [BW13]. These are defined over a *leveled multilinear group*, which is a sequence of groups $\mathbb{G}_1, \ldots, \mathbb{G}_\kappa$, each $\mathbb{G}_i$ of prime order $p > 2^\lambda$ and generated by g_i, equipped with bilinear maps ("pairings") $e_{i,j} \colon \mathbb{G}_i \times \mathbb{G}_j \to \mathbb{G}_{i+j}$, for $i + j \leq \kappa$. The bit-fixing PRF from [BW13] maps inputs from $\{0,1\}^n$ to an element of $\mathbb{G}_\kappa$ where $\kappa = n + 1$. A key is a tuple $k = (\alpha, d_{1,0}, d_{1,1}, \ldots, d_{n,0}, d_{n,1}) \in \mathbb{Z}_p^{2n+1}$ and the PRF is defined as

$$P(sk, x) := (g_{n+1})^{\alpha \prod_{i=1}^{n} d_{i,x_i}} \quad . \tag{1}$$

As noted in [BW13], the values $D_{i,j} := g_1^{d_{i,j}}$ could be made public, and inspection of the proof reveals that $A := g_2^\alpha$ could also be made public without affecting pseudorandomness. These values could be used to make the PRF output P publicly verifiable if we added one level in the group sequence, that is, set $\kappa := n + 2$. Then in order to verify that some $P \in \mathbb{G}_{n+1}$ equals $P(sk, x)$ as defined in (1), one could repeatedly apply the pairings to A and $D_{1,x_1}, \ldots, D_{n,x_n}$ to compute $(g_{n+2})^{\alpha \prod_{i=1}^{n} d_{i,x_i}} \in \mathbb{G}_\kappa$ and check whether this equals the pairing of P with g_1, which would lift P to $\mathbb{G}_{n+2}$.

Of course this shows that $P(sk, x)$ is not pseudorandom anymore after adding a level in the group hierarchy; however, it can serve as the proof for a related value in $\mathbb{G}_\kappa$. After adding an element $\gamma \in \mathbb{Z}_p$ to the secret key, we define the VRF value as $F(sk, x) := (g_\kappa)^{\gamma \cdot \alpha \prod_{i=1}^{n} d_{i,x_i}}$. The value $P(sk, x) = (g_{\kappa-1})^{\alpha \prod_{i=1}^{n} d_{i,x_i}}$ can now be used to check whether some $y \in \mathbb{G}_\kappa$ equals $F(sk, x)$: we add $C := g_1^\gamma$ to the public key and then have $e_{1,\kappa-1}(C, P(sk, x)) = (g_\kappa)^{\gamma \cdot \alpha \prod_{i=1}^{n} d_{i,x_i}} = F(sk, x)$. A nice side effect of this approach is that since our proof corresponds to the PRF value in [BW13], we can reuse their constrained keys to construct proofs. In particular for the circuit-constrained VRF this involves sophisticated techniques derived from [GGH+13c].

While this approach works for both the bit-fixing VRF and the circuit-constrained VRF, a drawback is that it requires an extra level in the group hierarchy. Somewhat surprisingly, we show that this is not necessary: we instantiate circuit-constrained VRFs using the same number of group levels as the Boneh-Waters circuit-constrained PRF and for the bit-fixing construction we even require one level less than [BW13].

The reason for this is that, as we show, the bit-fixing PRF can be constructed over a multilinear group with $\kappa = n - 1$ (rather than $\kappa = n + 1$ in [BW13]) and the circuit-constrained PRF can be constructed for $\kappa = n + \ell - 1$ (rather than $\kappa = n + \ell$ in [BW13]), where ℓ is the maximum depth of the circuits. This allows us to use the freed level for verification and preserve the function value of the PRF. We present these modified constrained PRFs and prove their security in Appendix A.

In Appendix B we show that, as for the constrained PRFs in [BW13], our constructions can be transferred from leveled multilinear groups to *graded encodings*, constructed by Garg, Gentry and Halevi [GGH13a], which can be viewed as "approximate" multilinear groups.

Complexity leveraging. Pseudorandomness of our VRFs can be reduced to the multilinear DDH assumption without any security loss when considering *selective* security. For this notion the adversary must decide on which value it wants to be challenged before receiving the public key. *Adaptive security* (where the adversary can make its challenge query at any point) can then be obtained generically via *complexity leveraging* [BB04a]: the reduction simply guesses beforehand which challenge value the

adversary will query. This leads to a security loss that is exponential in the input length, which must be compensated by increasing the parameters of the scheme.

Together with Konstantinov, Pietrzak and Rao [FKPR14], we recently showed that any *simple* reduction (that is, one which runs an adversary once without rewinding) from pseudorandomness of the Boneh-Waters constrained PRF to a non-interactive hardness assumption must incur a security loss that is exponential in the input length. Since constrained VRFs imply constrained PRFs, this also holds for our construction, meaning that our proofs using complexity leveraging are in some sense optimal.

Related work. VRFs have been constructed in bilinear groups by Lysyanskaya [Lys02] and Dodis [Dod03]. Based on Boneh-Boyen signatures [BB04b], Dodis and Yampolskiy [DY05] gave the first efficient scheme that is secure under a non-interactive assumption, but for small input spaces only. Hohenberger and Waters [HW10] proposed the first VRF for exponential-size domains without resorting to complexity leveraging or interactive assumptions. Boneh, Montgomery and Raghunathan [BMR10] achieve a similar result basing their construction on the Dodis-Yampolskiy VRF. Abdalla, Catalano and Fiore [ACF13] show connections of VRFs to identity-based key encapsulation, and also present a VRF with large input spaces. Evidence why VRFs are hard to construct is given by Brakerski, Goldwasser, Rothblum and Vaikuntanathan [BGRV09], who show that there is no black-box construction from one-way permutations, and Fiore and Schröder [FS12], showing that there is also none from trapdoor permutations. Variants of VRFs include simulatable VRFs [CL07] (where CRSs are allowed) and weak VRFs [BGRV09].

The concept of restricting keys for PRFs to subsets of their domains was concurrently introduced as constrained PRFs by Boneh and Waters [BW13], as delegatable PRFs by Kiayias et al. [KPTZ13], and as functional PRFs by Boyle et al. [BGI14]. The latter mention functional VRFs as an open problem.

An analogous notion for digital signatures, namely deriving signing keys that can only sign subsets of the message space was concurrently introduced by Boyle et al. [BGI14] as functional signatures and by Bellare and the author as policy-based signatures [BF14]. Since VRFs satisfy the definition of digital signatures, constrained VRFs immediately yield policy-based signatures (PBS) for the same classes of policies describing the constrained input (message) space. We note however that constraint-hiding constrained VRFs, which we construct in this paper, cannot satisfy the stronger of the two security definitions for PBS proposed in [BF14], which requires that the policy (constraint) can be extracted from a signature.

2 Preliminaries

Notation. If S is a finite set then $|S|$ denotes its size and $s \leftarrow\!\!\$\ S$ denotes picking an element uniformly from S and assigning it to s. For $n \in \mathbb{N}$ we let $[n] = \{1, \ldots, n\}$. We denote the security parameter by $\lambda \in \mathbb{N}$ and its unary representation by 1^λ. Algorithms are randomized unless otherwise indicated and "PT" stands for "polynomial-time" for both randomized and deterministic algorithms. We denote by $y := A(x_1, \ldots; \rho)$ the operation of running algorithm A on inputs $x_1, \ldots$ and coins ρ and assigning the output to y. By $y \leftarrow\!\!\$\ A(x_1, \ldots)$, we denote letting $y := A(x_1, \ldots; \rho)$ with ρ chosen at random. We denote by $[A(x_1, \ldots)]$ the set of points that have positive probability of being output by A on inputs $x_1, \ldots$.

Multilinear groups. The usefulness of groups with multilinear maps in which computing discrete logarithms is hard was first observed by Boneh and Silverberg [BS02]. It was only recently that candidates for leveled multilinear forms were proposed by Garg, Gentry and Halevi [GGH13a] and then by Coron, Lepoint and Tibouchi [CLT13]. Although these constructions implement *graded encodings*, which differ from multilinear groups, we present our results in the language of multilinear groups. These can then be transferred in a straightforward manner to graded encodings, as we show in Appendix B.

Leveled multilinear groups are generated by a group generator $\mathcal{G}$, which takes as input the security parameter 1^λ and $\kappa \in \mathbb{N}$, which determines the number of levels. $\mathcal{G}(1^\lambda, \kappa)$ outputs a sequence of groups $\vec{\mathbb{G}} = (\mathbb{G}_1, \ldots, \mathbb{G}_\kappa)$ of prime order $p > 2^\lambda$. We assume that the description of each group contains a canonical generator g_i. For all $i, j \geq 1$ with $i + j \leq \kappa$, there exists a bilinear map $e_{i,j} \colon \mathbb{G}_i \times \mathbb{G}_j \to \mathbb{G}_{i+j}$, which satisfies:

$$\forall a, b \in \mathbb{Z}_p : e_{i,j}\big(g_i^a, g_j^b\big) = (g_{i+j})^{a \cdot b} \ .$$

(We omit the indices i, j of the maps if they can be deduced from the context.)
The only hardness assumption we will make is the following:

Assumption 1. *The* κ-Multilinear Decisional Diffie-Hellman *(κ-MDDH) assumption states that, given* $(\mathbb{G}_1 \ldots, \mathbb{G}_\kappa)$ *obtained by running* $\mathcal{G}(1^\lambda, \kappa)$ *and* $g = g_1, g^{c_1}, \ldots, g^{c_{\kappa+1}}$ *for* $c_1, \ldots, c_{\kappa+1} \leftarrow_\$ \mathbb{Z}_p$*, it is hard to distinguish* $g_\kappa^{\prod_{j \in [\kappa+1]} c_j} \in \mathbb{G}_\kappa$ *from a random group element in* $\mathbb{G}_\kappa$*.*

Circuits. Our treatment of circuits follows that by Boneh and Waters [BW13], who adapt the model of Bellare et al. [BHR12]. They consider boolean circuits with a single output gate and require that circuits are *layered* (where a gate at level j receives its inputs from wires at level $j-1$) and *monotonic* in that they only contain AND and OR gates. This is without loss of generality, since an arbitrary circuit can be transformed into a layered monotonic circuit of polynomially related size.

Definition 1. *A* circuit *is a 5-tuple* $f = (n, q, A, B, \texttt{GateType})$*, where* n *is the number of inputs and* q *is the number of gates. Wires are associated with the set* $[n + q] = \{1, \ldots, n + q\}$*, where* $\{1, \ldots, n\}$ *are the input wires and* $n + q$ *is the output wire. Gates are labeled by the same index as their outgoing wire, we thus define* $\mathsf{Gates} := \{n + 1, \ldots, n + q\}$*.*

The function $A \colon \mathsf{Gates} \to [n + q]$ *maps a gate* w *to its first incoming wire* $A(w)$ *and* $B \colon \mathsf{Gates} \to [n + q]$ *maps a gate* w *to its second incoming wire* $B(w)$*. We require* $w > B(w) > A(w)$*. The function* $\texttt{GateType} \colon \mathsf{Gates} \to \{\text{AND}, \text{OR}\}$ *specifies whether a gate is an AND or an OR gate.*

The function $\texttt{depth}(w)$ *maps a wire to the length of the shortest path to an input wire plus 1; in particular for* $w \in [n]$ *we have* $\texttt{depth}(w) = 1$*. Moreover, a circuit is* layered *if for all* $w \in \mathsf{Gates}$: $\texttt{depth}(A(w)) = \texttt{depth}(B(w)) = \texttt{depth}(w) - 1$*. We let* $f(x)$ *denote the evaluation of the circuit* f *on input* $x \in \{0, 1\}^n$ *and let* $f_w(x)$ *denote the value of wire* w *of the circuit on input* x*.*

3 Constrained Verifiable Random Functions

We extend the definition of constrained pseudorandom functions (PRF), defined by Boneh and Waters [BW13] to constrained verifiable random functions (VRF). A constrained PRF allows one to evaluate a keyed function $F \colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ and defines an algorithm that given a key $k \in \mathcal{K}$ and a set $S \subseteq \mathcal{X}$ derives a key k_S with which one can only evaluate F on points $x \in S$. It is set up w.r.t. a set system $\mathcal{S} \subseteq 2^{\mathcal{X}}$, defining the sets for which constrained keys can be derived.

For VRFs, in addition to a (secret) key, the setup algorithm outputs a public key pk. Given a constrained secret key sk_S derived from sk for a set $S \in \mathcal{S}$ and an input $x \in S$, the algorithm Prove computes the value $y = F(sk, x)$ (like the algorithm eval in [BW13]). It moreover outputs a proof π for the fact that $F(sk, x) = y$, which can be verified w.r.t. pk via an algorithm Verify.

We require that a constrained VRF satisfies the following properties: *Provability* ensures completeness of the scheme: running Prove on a constrained key outputs the correct function value and a proof that passes verification. *Uniqueness* guarantees soundness of the proofs: for any (possibly maliciously computed) value pk and every $x \in \mathcal{X}$ there exists at most one $y \in \mathcal{Y}$ for which $\mathsf{Verify}(pk, x, y, \pi) = 1$ for some π. Compared to PRFs, *pseudorandomness* should also hold against adversaries that obtain the public key and proofs for input points in addition to function values and constrained keys of their choice.

Finally, we consider an additional privacy or anonymity notion, which ensures that proofs do not reveal anything about the constrained key used to compute them: proofs computed with a constrained key should be distributed like proofs computed with the actual secret key. Note that this notion would not be meaningful for constrained PRFs or (standard) VRFs: a constrained key for a PRF is only used to evaluate F, so by definition, different constrained keys yield the same output; and for standard VRFs all proofs are computed with the same key.

Definition. Let $F\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a function computable in polynomial time, where $\mathcal{K}$ is the key space, $\mathcal{X}$ is the domain and $\mathcal{Y}$ the range (which may all be parametrized by the security parameter λ). F is said to be a *constrained verifiable random function* w.r.t. a set system $\mathcal{S} \subseteq 2^{\mathcal{X}}$ if there exists a *constrained-key space* $\mathcal{K}'$, a *proof space* $\mathcal{P}$ and PT algorithms Setup, $\mathsf{Constrain}$, Prove and Verify:

- $\mathsf{Setup}(1^\lambda)$ outputs a pair of keys (pk, sk).
- $\mathsf{Constrain}(sk, S)$, on input a secret key and a set $S \in \mathcal{S}$, outputs a constrained key $sk_S \in \mathcal{K}'$.
- $\mathsf{Prove}(sk_S, x)$ outputs a pair $(y, \pi) \in \mathcal{Y} \times \mathcal{P} \cup \{(\bot, \bot)\}$ of a function value and a proof.
- $\mathsf{Verify}(pk, x, y, \pi)$ verifies that $y = F(sk, x)$ using proof π, outputting a value in $\{0, 1\}$.

We require the following properties:

Provability. For all $\lambda \in \mathbb{N}$, all $(pk, sk) \in [\mathsf{Setup}(1^\lambda)]$, all $S \in \mathcal{S}$, all $sk_S \in [\mathsf{Constrain}(sk, S)]$, all $x \in \mathcal{X}$ and $(y, \pi) \in [\mathsf{Prove}(sk_S, x)]$ it holds that:

- If $x \in S$ then $y = F(sk, x)$ and $\mathsf{Verify}(pk, x, y, \pi) = 1$
- If $x \notin S$ then $(y, \pi) = (\bot, \bot)$

Uniqueness. For all $\lambda \in \mathbb{N}$, all pk, all $x \in \mathcal{X}$, $y_0, y_1 \in \mathcal{Y}$ and $\pi_0, \pi_1 \in \mathcal{P}$ one of the following holds:

- $y_0 = y_1$,
- $\mathsf{Verify}(pk, x, y_0, \pi_0) = 0$, or
- $\mathsf{Verify}(pk, x, y_1, \pi_1) = 0$,

that is, for every x there is at most one value y for which there exists a proof that $F(sk, x) = y$.

Constraint-hiding. This notion ensures that the proof does not reveal which key was used to create it. We require that there exist a PT algorithm $P\colon \mathcal{K} \times \mathcal{X} \to \mathcal{P}$, such that for all $\lambda \in \mathbb{N}$, all

$(pk, sk) \in [\mathsf{Setup}(1^\lambda)]$, all $S \in \mathcal{S}$, all $sk_S \in [\mathsf{Constrain}(sk, S)]$ and all $x \in S$ the following holds: the second output, π, of $\mathsf{Prove}(sk_S, x)$ and the output of $P(sk, x)$ are distributed identically.

Pseudorandomness. Consider the following experiment $\mathbf{Exp}_b^{\mathrm{pr}}(\lambda)$ for $\lambda \in \mathbb{N}$ and $b \in \{0,1\}$:

- Generate $(pk, sk) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$.
- Initialize sets C and V to $\emptyset$, where V will contain the points the adversary can evaluate and C records the points at which the adversary queried a challenge. Moreover, initialize an empty list R indexed by the set $\mathcal{X}$, used to store random values.
- Run the adversary on pk and provide the following oracles:

  CONSTRAIN: On input $S \in \mathcal{S}$, if $S \cap C = \emptyset$, return $sk_S \leftarrow\!\!\$\ \mathsf{Constrain}(sk, S)$ and set $V := V \cup S$; else return $\perp$.

  PROVE: Given $x \in \mathcal{X}$, if $x \notin C$, return $F((sk, x), P(sk, x))$ and set $V := V \cup \{x\}$; else return $\perp$.

  CHALLENGE: On input $x \in \mathcal{X}$, if $x \in V$ then return $\perp$. Else set $C := C \cup \{x\}$ and do the following. If $b = 0$ then return $F(sk, x)$; if $b = 1$ then return a consistent random value from $\mathcal{Y}$, that is, return $R[x]$ if $x \in C$ and otherwise choose $y \leftarrow\!\!\$\ \mathcal{Y}$, set $R[x] := y$ and return y.
- Let $b' \in \{0,1\}$ be the adversary's final output, which we define as the output of the experiment.

A constrained VRF is *pseudorandom* if the function $\left|\Pr[\mathbf{Exp}_1^{\mathrm{pr}}(\lambda) = 1] - \Pr[\mathbf{Exp}_0^{\mathrm{pr}}(\lambda) = 1]\right|$ is negligible in λ for all PT adversaries $\mathcal{A}$.

Note that by the constraint-hiding property an oracle to obtain Prove evaluations under constrained keys unknown to the adversary would be redundant. In our security proofs we will only allow the adversary to query its challenge oracle once. This restricted notion however implies the notion defined above via a standard hybrid argument.

4 Bit-Fixing VRF

In our first construction constrained keys can be derived for any "bit-fixing" set. Such a set is defined by a value $\mathbf{v} \in \{0, 1, ?\}^n$ as the set of all $x \in \{0,1\}^n$ that match $\mathbf{v}$ at all positions where $\mathbf{v}$ is different from '?':

$$S_{\mathbf{v}} := \left\{x \in \{0,1\}^n \;\middle|\; \forall i \in [n] : x_i = \mathbf{v}_i \vee \mathbf{v}_i = ?\right\}$$

The set system for our constrained VRF is then defined as $\mathcal{S} := \left\{S_{\mathbf{v}} \subseteq \{0,1\}^n \,|\, \mathbf{v} \in \{0, 1, ?\}^n\right\}$.

We show how to add verifiability to the bit-fixing PRF by Boneh and Waters [BW13], which has domain $\mathcal{X} = \{0,1\}^n$ and where keys can be derived for $S_{\mathbf{v}}$ for every $\mathbf{v} \in \{0, 1, ?\}^n$. As discussed in the introduction, the idea is to use one extra level of the group hierarchy for verification: the element that was the PRF value now serves as proof and the VRF value will live one group level above. Verification is done using the pairings to check consistency. For their bit-fixing PRF, Boneh and Waters define

$$F_{\mathrm{PRF}} \colon \mathcal{K} \times \{0,1\}^n \to \mathbb{G}_{n+1}, \quad \left((\alpha, \{d_{i,\beta}\}_{i\in[n],\,\beta\in\{0,1\}}), x\right) \mapsto (g_{n+1})^{\alpha \prod_{i\in[n]} d_{i,x_i}} \ .$$

In Appendix A.1 we show that $n-1$ group levels suffice when one defines the PRF value as $F'_{\mathrm{PRF}}(sk, x) = (g_{n-1})^{\prod_{i\in[n]} d_{i,x_i}}$. We use this value as the proof in our VRF construction and the same constrained keys for both the modified PRF and the VRF. In order to provide verifiability, we add back one level; the last group in our hierarchy is thus $\mathbb{G}_n$, which is one level below the one of the Boneh-Waters PRF.[1]

[1] If we wanted a VRF with the same function values as the Boneh-Waters PRF, it would suffice to set up $\kappa = n + 1$

4.1 Construction

$\underline{\mathsf{Setup}(1^\lambda, 1^n)}$: On input the security parameter λ and the input length n, the setup runs $\mathcal{G}(1^\lambda, n)$ to compute a sequence of groups $\vec{\mathbb{G}} = (\mathbb{G}_1, \ldots, \mathbb{G}_n)$ of prime order p, with generators $g_1, \ldots, g_n$, of which we let $g := g_1$. It chooses $\gamma \leftarrow\!\!\$\ \mathbb{Z}_p$ and $(d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}) \leftarrow\!\!\$\ \mathbb{Z}_p^2$ uniformly at random and sets $C := g^\gamma$ and $D_{i,\beta} := g^{d_{i,\beta}}$ for $i \in [n]$ and $\beta \in \{0,1\}$. The VRF public and secret key are defined as

$$pk := \big(\vec{\mathbb{G}} = (\mathbb{G}_1, \ldots, \mathbb{G}_n), C, \{D_{i,\beta}\}_{i\in[n],\,\beta\in\{0,1\}}\big)$$
$$sk := \big(pk, \gamma, \{d_{i,\beta}\}_{i\in[n],\,\beta\in\{0,1\}}\big)$$

The domain is $\mathcal{X} = \{0,1\}^n$, the range of the function is $\mathcal{Y} = \mathbb{G}_n$ and proofs are in $\mathbb{G}_{n-1}$. The function value and the proof for input $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ are defined as

$$F(sk, x) := {g_n}^{\gamma \prod_{i\in[n]} d_{i,x_i}} \qquad\qquad P(sk, x) := (g_{n-1})^{\prod_{i\in[n]} d_{i,x_i}}$$

$\underline{\mathsf{Verify}(pk, x, y, \pi)}$: To verify a tuple $(x, y, \pi) \in \{0,1\}^n \times \mathbb{G}_n \times \mathbb{G}_{n-1}$ w.r.t. public key $pk = (\vec{\mathbb{G}}, C, \{D_{i,\beta}\})$, compute $D(x) := {g_n}^{\prod_{i\in[n]} d_{i,x_i}}$ by applying the bilinear maps to $(D_{1,x_1}, \ldots, D_{n,x_n})$ and output 1 if the following equations are satisfied:

$$e(g, \pi) = D(x) \qquad\qquad e(C, \pi) = y$$

$\underline{\mathsf{Constrain}(sk, \mathbf{v})}$: Note that from a proof $P(sk, x)$, by pairing it with the public-key element C, one can compute $F(sk, x) = e(C, P(sk, x))$. It suffices thus that a constrained key lets us construct $P(sk, x)$.

The algorithm takes as input sk and a vector $\mathbf{v} \in \{0, 1, ?\}^n$ describing the constrained domain $S_\mathbf{v} := \{x \in \{0,1\}^n \,|\, \forall i \in [n] : x_i = \mathbf{v}_i \vee \mathbf{v}_i = ?\}$. Let $V := \{i \in [n] \,|\, \mathbf{v}_i \neq ?\}$ be the set of indices for which the input bit is fixed to 0 or 1. Return $sk_\mathbf{v} := (pk, k_\mathbf{v})$, with $k_\mathbf{v}$ defined as follows:

- If $|V| > 1$ then compute $k_\mathbf{v} := (g_{|V|-1})^{\prod_{i\in V} d_{i,\mathbf{v}_i}}$.
- If $V = \{j\}$ then set $k_\mathbf{v} := d_{j,\mathbf{v}_j}$.

(If $V = \emptyset$ then return sk, from which $\mathsf{Prove}(sk, x)$ simply computes $F(sk, x)$ and $P(sk, x)$.)

$\underline{\mathsf{Prove}(sk_\mathbf{v}, x)}$: Again let $V := \{i \in [n] \,|\, \mathbf{v}_i \neq ?\}$ and let $\overline{V} := \{i \in [n] \,|\, \mathbf{v}_i = ?\}$ be its complement. If $x_i \neq \mathbf{v}_i$ for some $i \in V$ then return $(\perp, \perp)$; else apply the bilinear maps to $\{D_{i,x_i}\}_{i\in\overline{V}}$ to compute

$$D_{\overline{V}}(x) := (g_{|\overline{V}|})^{\prod_{i\in\overline{V}} d_{i,x_i}} \quad .$$

- If $|V| > 1$, set $P(sk, x) := e(D_{\overline{V}}(x), k_\mathbf{v}) = e\big((g_{|\overline{V}|})^{\prod_{i\in\overline{V}} d_{i,x_i}}, (g_{|V|-1})^{\prod_{i\in V} d_{i,\mathbf{v}_i}}\big) = (g_{n-1})^{\prod_{i\in[n]} d_{i,x_i}}$.
- If $V = \{j\}$, set $P(sk, x) := D_{\overline{V}}(x)^{k_\mathbf{v}} = \big((g_{|\overline{V}|})^{\prod_{i\in\overline{V}} d_{i,x_i}}\big)^{d_{j,\mathbf{v}_j}} = (g_{n-1})^{\prod_{i\in[n]} d_{i,x_i}}$.

Finally, compute $e\big(C, P(sk, x)\big) = e\big(g^\gamma, (g_{n-1})^{\prod_{i\in[n]} d_{i,x_i}}\big) = F(sk, x)$.

---

group levels, lift the domain of F from $\mathbb{G}_n$ to $\mathbb{G}_{n+1}$, that of P from $\mathbb{G}_{n-1}$ to $\mathbb{G}_n$, and define the public-key element $C := g_2^\gamma$ (instead of g^γ). The secret key element γ would then correspond to α in [BW13].

4.2 Properties

Provability. When $(pk = (\vec{\mathbb{G}}, C, \{D_{i,\beta}\}), sk) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$ then from the definition of F and P it follows immediately that for all $x \in \{0,1\}^n$: $e(g, P(sk,x)) = {g_n}^{\prod_{i\in[n]} d_{i,x_i}} = D(x)$ and $e(C, P(sk,x)) = {g_n}^{\gamma \prod_{i\in[n]} d_{i,x_i}} = F(sk,x)$. We have thus $\mathsf{Verify}(pk, x, F(sk,x), P(sk,x)) = 1$.

Moreover, given a constrained key $sk_{\mathbf{v}}$ derived for a vector $\mathbf{v} \in \{0,1,?\}^n$ and $x \in \{0,1\}^n$ with $x_i = \mathbf{v}_i$ or $\mathbf{v}_i = ?$ for all i, it follows by inspection that $\mathsf{Prove}(sk_{\mathbf{v}}, x)$ computes $(F(sk,x), P(sk,x))$, which we showed satisfy the verification equations.

Uniqueness. Consider a public key $pk = (\vec{\mathbb{G}}, C, \{D_{i,\beta}\}_{i\in[n],\,\beta\in\{0,1\}})$, with $C \in \mathbb{G}_1$ and $D_{i,\beta} \in \mathbb{G}_1$, a value $x \in \{0,1\}^n$ and values $(y_0, \pi_0), (y_1, \pi_1) \in \mathbb{G}_n \times \mathbb{G}_{n-1}$ that satisfy $\mathsf{Verify}(pk, x, y_\beta, \pi_\beta) = 1$, for $\beta \in \{0,1\}$. It suffices to show that $y_0 = y_1$.

Let $\gamma, d_{i,\beta} \in \mathbb{Z}_p$ be such that $C = g^\gamma$ and $D_{i,\beta} = g^{d_{i,\beta}}$ for all i, β. The fist verification equation yields $e(g, \pi_\beta) = {g_n}^{\prod_{i\in[n]} d_{i,x_i}}$, which by the properties of the bilinear map e implies that $\pi_\beta = (g_{n-1})^{\prod_{i\in[n]} d_{i,x_i}}$ for $\beta \in \{0,1\}$. The second equation yields $y_\beta = e(C, \pi_\beta) = e\big(g^\gamma, (g_{n-1})^{\prod_{i\in[n]} d_{i,x_i}}\big)$ for both $\beta \in \{0,1\}$, which implies $y_0 = {g_n}^{\gamma \prod_{i\in[n]} d_{i,x_i}} = y_1$.

Constraint-hiding. The proof algorithm P maps $sk = (\gamma, \{d_{i,\beta}\}_{i\in[n],\,\beta\in\{0,1\}})$ and $x \in \{0,1\}^n$ to $P(sk,x) := (g_{n-1})^{\prod_{i\in[n]} d_{i,x_i}}$. Since by provability, this is precisely the value that $\mathsf{Prove}(sk_{\mathbf{v}}, x)$ outputs for any constraint $\mathbf{v}$ and any x satisfying $\mathbf{v}$, the constraint-hiding property follows immediately.

4.3 Proof of Pseudorandomness

Theorem 1. *If there exists a PT adversary $\mathcal{A}$ that makes one challenge query and breaks pseudorandomness of the above n-bit-input bit-fixing VRF with advantage $\epsilon(\lambda)$ then there exists a PT algorithm $\mathcal{B}$ that breaks the n-Multilinear Decisional Diffie-Hellman assumption with advantage $2^{-n} \cdot \epsilon(\lambda)$.*

Proof. Without loss of generality, we assume that when x^* is $\mathcal{A}$'s challenge query then $\mathcal{A}$ never queries constrained keys that could evaluate x^*, nor its PROVE oracle on x^*. We construct $\mathcal{B}$, which receives an n-MDDH challenge consisting of a group-sequence description $\vec{\mathbb{G}}$ and elements $g = g_1, g^{c_1}, \ldots, g^{c_{n+1}}$ and T, which is either ${g_n}^{\prod_{i\in[n+1]} c_j}$ or a random element from $\mathbb{G}_n$. $\mathcal{B}$ picks a random value $x^* \leftarrow\!\!\$\ \{0,1\}^n$, which it hopes will be $\mathcal{A}$'s challenge query, and $z_1, \ldots, z_n \leftarrow\!\!\$\ \mathbb{Z}_p$ and sets

$$D_{i,\beta} := \begin{cases} g^{c_i} & \text{if } x^*_i = \beta \\ g^{z_i} & \text{if } x^*_i \neq \beta \end{cases} \qquad \text{for } i \in [n],\ \beta \in \{0,1\},$$

which implicitly defines $d_{i,x^*_i} := c_i$ and $d_{i,\overline{x}^*_i} := z_i$ (with $\overline{x}^*_i$ denoting $1 - x^*_i$). It also defines $\gamma := c_{n+1}$ by setting $C := g^{c_{n+1}}$. $\mathcal{B}$ then runs $\mathcal{A}$ on input the public key $(\vec{\mathbb{G}}, C, \{D_{i,\beta}\}_{i\in[n],\,\beta\in\{0,1\}})$, which is distributed as in the real scheme.

Constrain queries. Suppose $\mathcal{A}$ queries a secret key for $\mathbf{v} \in \{0,1,?\}^n$. Let $V := \{i \in [n] \,|\, \mathbf{v}_i \neq ?\}$ be the set of indices that $\mathbf{v}$ fixes. $\mathcal{B}$ selects $j \in V$ such that $\mathbf{v}_j \neq x^*_j$. If no such j exists then the key could be used to evaluate $F(sk, x^*)$, meaning $\mathcal{B}$'s guess was wrong, as we assumed $\mathcal{A}$ would not make such a query. In this case $\mathcal{B}$ aborts outputting a random guess $b' \leftarrow\!\!\$\ \{0,1\}$.

If $|V| = 1$ then $V = \{j\}$, thus $\mathcal{B}$ knows z_j with $D_{j,\mathbf{v}_j} = D_{j,\overline{x}^*_i} = g^{z_j}$ and sets $k_{\mathbf{v}} := z_j$. If $|V| > 1$ then by repeatedly applying the bilinear maps to the values $\{D_{i,\mathbf{v}_i}\}_{i\in V\setminus\{j\}}$, it computes $(g_{|V|-1})^{\prod_{i\in V\setminus\{j\}} d_{i,\mathbf{v}_i}}$ and raises this value to $z_j = d_{j,\mathbf{v}_j}$ to compute $k_{\mathbf{v}} := (g_{|V|-1})^{\prod_{i\in V} d_{i,\mathbf{v}_i}}$. $\mathcal{B}$ answers the query with $(pk, k_{\mathbf{v}})$.

Prove queries. Since $P(sk, x)$ is identical to a key for $\mathbf{v} = x$, this value can be computed as for the constrained-key query above. $F(sk, x)$ is computed by pairing it with C.

Challenge query. If $\mathcal{A}$'s challenge query is different from x^* then $\mathcal{B}$ aborts outputting a random guess $b' \leftarrow\!\!\$\ \{0,1\}$. Otherwise, it outputs T as a response to the query. If $T = g_n^{\prod_{i\in[n+1]} c_j}$ then $T = g_n^{\gamma \prod_{i\in[n]} d_{i,x_i^*}} = F(sk, x^*)$. When $\mathcal{A}$ outputs a guess b' then $\mathcal{B}$, if it has not aborted, outputs the same guess b'.

Success probability. We analyze the probability that $\mathcal{B}$ wins the MDDH game. Let $\texttt{abort}$ denote the event that $\mathcal{B}$ aborts during the simulation. $\mathcal{B}$ aborts if and only if $\mathcal{A}$ queries its CHALLENGE oracle on a value different from x^* (as $\mathcal{A}$ does not make any "illegal" queries, $\mathcal{B}$ only aborts during CONSTRAIN and PROVE queries if its guess of x^* was wrong). We therefore have $\Pr[\texttt{abort}] = 1 - 2^{-n}$. Moreover, if $\mathcal{B}$ aborts then it outputs a random bit, thus yielding $\Pr[\mathcal{B} \text{ wins} \,|\, \texttt{abort}] = \frac{1}{2}$. If $\mathcal{B}$ does not abort then it wins with the same probability as $\mathcal{A}$ (since $\mathcal{A}$'s success is independent of $\mathcal{B}$'s guess of x^*), whose advantage is $\epsilon(\lambda)$, thus $\Pr[\mathcal{B} \text{ wins} \,|\, \overline{\texttt{abort}}] = \frac{1}{2} + \epsilon(\lambda)$. Together, we have

$$\begin{aligned}\Pr[\mathcal{B} \text{ wins}] &= \Pr[\mathcal{B} \text{ wins} \,|\, \texttt{abort}] \cdot \Pr[\texttt{abort}] + \Pr[\mathcal{B} \text{ wins} \,|\, \overline{\texttt{abort}}] \cdot \Pr[\overline{\texttt{abort}}] \\ &= \tfrac{1}{2} \cdot (1 - 2^{-n}) + \left(\tfrac{1}{2} + \epsilon(\lambda)\right) \cdot 2^{-n} \;=\; \tfrac{1}{2} + 2^{-n} \cdot \epsilon(\lambda) \;,\end{aligned}$$

which shows that $\mathcal{B}$'s advantage in breaking n-MDDH is $\epsilon'(\lambda) = 2^{-n} \cdot \epsilon(\lambda)$. □

5 Circuit-Constrained VRF

Consider a polynomial-size circuit f as in Definition 1. Our second VRF construction allows us to derive a constrained key sk_f enabling function evaluations and proof computations for exactly those values x, for which $f(x) = 1$. Letting $\mathcal{C}$ be the set of all polynomial-size circuits, we have

$$\mathcal{S} := \{S_f \subseteq \{0,1\}^n \,|\, f \in \mathcal{C}\} \;, \qquad \text{with} \;\; S_f := \{x \in \{0,1\}^n \,|\, f(x) = 1\} \;.$$

Our circuit-constrained VRF is derived from the Boneh-Waters PRF [BW13] for the same set system. Their PRF values are in $\mathbb{G}_\kappa$ with $\kappa = n + \ell$, where ℓ is the maximum depth of the supported circuits. In Appendix A.2 we show that their PRF construction can be modified and defined over a group sequence with $\kappa = n + \ell - 1$, by shifting the PRF value and elements of the constrained key down by one level. Pseudorandomness then follows from $(n + \ell - 1)$-MDDH.

For our constrained VRF we define the proofs as the values of the modified PRF in $\mathbb{G}_{n+\ell-1}$ (so proofs can be constructed using the constrained keys of the modified PRF), then add back one level in the group hierarchy and define the function values in $\mathbb{G}_{n+\ell}$ as pairings of the proof with an additional public-key element g^γ. The Boneh-Waters PRF is defined as $F_{\text{PRF}}(k, x) := g_\kappa^{\alpha' \prod_{i\in[n]} d_{i,x_i}}$, where the key k consists of α' and the elements $d_{i,\beta}$. Our VRF values can be seen as the same but with α' split into α and γ, thus $F(sk, x) := g_\kappa^{\alpha\cdot\gamma \prod_{i\in[n]} d_{i,x_i}}$. The proof is the same value without γ and lives one level below the function value: $P(sk, x) := (g_{\kappa-1})^{\alpha \prod_{i\in[n]} d_{i,x_i}}$.

5.1 Construction

$\underline{\mathsf{Setup}(1^\lambda, 1^n, 1^\ell)}$: On input the security parameter λ, the bit length n and the maximum depth ℓ of the circuits, Setup does the following: Run $\mathcal{G}(1^\lambda, \kappa)$ with $\kappa := n + \ell$ to obtain a sequence of groups

$\vec{\mathbb{G}} = (\mathbb{G}_1, \ldots, \mathbb{G}_\kappa)$ of prime order p, with generators $g = g_1, \ldots, g_\kappa$. Choose secret-key values $\alpha, \gamma \leftarrow\!\!\$\ \mathbb{Z}_p$ and $(d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}) \leftarrow\!\!\$\ \mathbb{Z}_p^2$ and set $A := g_\ell^\alpha$, $C := g^\gamma$ and $D_{i,\beta} := g^{d_{i,\beta}}$ for $i \in [n]$ and $\beta \in \{0,1\}$. The VRF public key pk is defined as the group sequence $\vec{\mathbb{G}}$ and $\big(A, C, \{D_{i,\beta}\}_{i\in[n],\,\beta\in\{0,1\}}\big)$. The secret key sk consists of the public key as well as $\big(\alpha, \gamma, \{d_{i,\beta}\}_{i\in[n],\,\beta\in\{0,1\}}\big)$.

We define the domain as $\mathcal{X} := \{0,1\}^n$, the range as $\mathcal{Y} := \mathbb{G}_\kappa$, and the proof space as $\mathcal{P} := \mathbb{G}_{\kappa-1}$. On input $x = (x_1, \ldots, x_n) \in \mathcal{X}$, the function value and the proof are defined as

$$F(sk, x) := {g_\kappa}^{\alpha\cdot\gamma \prod_{i\in[n]} d_{i,x_i}} \qquad\qquad P(sk, x) := (g_{\kappa-1})^{\alpha \prod_{i\in[n]} d_{i,x_i}}$$

$\underline{\mathsf{Verify}(pk, x, y, \pi)}$: Given a public key $pk = (\vec{\mathbb{G}}, A, C, \{D_{i,\beta}\}_{i,\beta})$ and $(x, y, \pi) \in \{0,1\}^n \times \mathbb{G}_\kappa \times \mathbb{G}_{\kappa-1}$, first compute $D(x) := {g_n}^{\prod_{i\in[n]} d_{i,x_i}}$ by applying the bilinear maps to $(D_{1,x_1}, \ldots, D_{n,x_n})$ and return 1 if the following equations hold (and return 0 otherwise):

$$e(g, \pi) = e(A, D) \qquad\qquad e(C, \pi) = y$$

$\underline{\mathsf{Constrain}(sk, f = (n, q, A, B, \texttt{GateType}))}$: On input the secret key and a circuit description f, with n input wires, q gates (labeled from $n+1$ to $n+q$), and the wire $n+q$ designated as output wire, the constrain algorithm does the following:

Choose $r_1, \ldots, r_{n+q-1} \leftarrow\!\!\$\ \mathbb{Z}_p$ and set $r_{n+q} := \alpha$. For every wire w generate a key component K_w, whose structure depends on the type of the wire: input wire, OR gate, or AND gate.

Input wire: If $w \in [n]$, it corresponds to the w-th input and the key component is

$$K_w := g^{r_w \cdot d_{w,1}} \quad .$$

OR gate: If $w \in \mathsf{Gates}$ with $\texttt{GateType}(w) = \text{OR}$ and $\texttt{depth}(w) = j$ then choose $a_w, b_w \leftarrow\!\!\$\ \mathbb{Z}_p$ and compute the following key components:

$$K_{w,1} := g^{a_w} \qquad K_{w,2} := g^{b_w} \qquad K_{w,3} := (g_{j-1})^{r_w - a_w \cdot r_{A(w)}} \qquad K_{w,4} := (g_{j-1})^{r_w - b_w \cdot r_{B(w)}}$$

AND gate: If $w \in \mathsf{Gates}$ with $\texttt{GateType}(w) = \text{AND}$ and $\texttt{depth}(w) = j$ then choose $a_w, b_w \leftarrow\!\!\$\ \mathbb{Z}_p$ and compute the following key components:

$$K_{w,1} := g^{a_w} \qquad\qquad K_{w,2} := g^{b_w} \qquad\qquad K_{w,3} := (g_{j-1})^{r_w - a_w \cdot r_{A(w)} - b_w \cdot r_{B(w)}}$$

The constrained key sk_f consists of these components for all $n+q$ wires together with the circuit description f and the public key pk.

$\underline{\mathsf{Prove}(sk_f, x)}$: Given a constrained key sk_f for circuit $f = (n, q, A, B, \texttt{GateType})$ and input $x \in \{0,1\}^n$, if $f(x) = 0$, return $(\perp, \perp)$. Otherwise, evaluate the circuit level by level starting from the input wires. For every wire w that evaluates to 1, compute the value $P_w = (g_{n+j-1})^{r_w \prod_{i\in[n]} d_{i,x_i}}$, where $j = \texttt{depth}(w)$. Note that since $r_{n+q} = \alpha$, we have $P_{n+q} = P(sk, x)$, from which we can then compute $F(sk, x)$ by pairing it with C. For every wire we distinguish the following cases:

Input wire: For $w \in [n]$ we only consider those w for which $x_w = f_w(x) = 1$. Repeatedly apply the bilinear maps to the values $\{D_{i,x_i}\}_{i\neq w}$ to compute $(g_{n-1})^{\prod_{i\in[n]\setminus\{w\}} d_{i,x_i}}$ and pair it with $K_w = g^{r_w \cdot d_{w,1}}$ to obtain $P_w = {g_n}^{r_w \prod_{i\in[n]} d_{i,x_i}}$.

OR gate: Let $w \in \mathsf{Gates}$ be such that $f_w(x) = 1$ and $\mathtt{GateType}(w) = \text{OR}$ and let $j = \mathtt{depth}(w)$. Define $D(x) := g_n^{\prod_{i\in[n]} d_{i,x_i}}$, which can be computed from the set $\{D_{i,x_i}\}_{i\in[n]}$.

If $f_{A(w)}(x) = 1$ then compute:

$$e(P_{A(w)}, K_{w,1}) \cdot e(K_{w,3}, D(x))$$
$$= e\big((g_{n+j-2})^{r_{A(w)} \prod_i d_{i,x_i}}, g^{a_w}\big) \cdot e\big((g_{j-1})^{r_w - a_w \cdot r_{A(w)}}, g_n^{\prod_i d_{i,x_i}}\big) = (g_{j+n-1})^{r_w \prod_i d_{i,x_i}} = P_w \ .$$

Otherwise, we must have $f_{B(w)}(x) = 1$, so compute:

$$e(P_{B(w)}, K_{w,2}) \cdot e(K_{w,4}, D(x)) = (g_{j+n-1})^{r_w \prod_i d_{i,x_i}} = P_w \ .$$

AND gate: Let $w \in \mathsf{Gates}$ be such that $f_w(x) = 1$, $\mathtt{GateType}(w) = \text{AND}$ and $\mathtt{depth}(w) = j$. We have $f_{A(w)} = f_{B(w)} = 1$ and with $D(x)$ as above we compute:

$$e(P_{A(w)}, K_{w,1}) \cdot e(P_{B(w)}, K_{w,2}) \cdot e(K_{w,3}, D(x))$$
$$= e\big((g_{n+j-2})^{r_{A(w)} \prod_i d_{i,x_i}}, g^{a_w}\big) \cdot e\big((g_{n+j-2})^{r_{B(w)} \prod_i d_{i,x_i}}, g^{b_w}\big) \cdot e\big((g_{j-1})^{r_w - a_w \cdot r_{A(w)} - b_w \cdot r_{B(w)}}, g_n^{\prod_i d_{i,x_i}}\big)$$
$$= (g_{j+n-1})^{r_w \prod_i d_{i,x_i}} = P_w \ .$$

Evaluating level by level all wires w for which $f_w(x) = 1$, we arrive at $P_{n+q} = (g_{n+\ell-1})^{\alpha \prod_i d_{i,x_i}} = P(sk, x)$, from which we compute

$$e(C, P(sk, x)) = e\big(g^\gamma, (g_{\kappa-1})^{\alpha \prod_i d_{i,x_i}}\big) = F(sk, x)$$

and output $\big(F(sk, x), P(sk, x)\big)$.

5.2 Properties

Provability. When $\big(pk = (\vec{\mathbb{G}}, A, C, \{D_{i,\beta}\}), sk\big) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda, 1^n, 1^\ell)$ then from the definition of F and P it follows that for all $x \in \{0,1\}^n$: $e\big(g, P(sk, x)\big) = e\big(A, D(x)\big)$ and $e\big(C, P(sk, x)\big) = F(sk, x)$.

Moreover, given a constrained key sk_f derived from sk for a depth-ℓ circuit f and $x \in \{0,1\}^n$ with $f(x) = 1$, we see that when running the Prove algorithm, the value computed for every depth-j gate w for which $f_w(x) = 1$ is $P_w = (g_{n+j-1})^{r_w \prod_{i\in[n]} d_{i,x_i}}$. Since the value r_{n+q} for the output gate was set as $r_{n+q} := \alpha$, Prove outputs $(g_{n+\ell-1})^{\alpha \prod_{i\in[n]} d_{i,x_i}} = P(sk, x)$ and $e\big(C, P(sk, x)\big) = F(sk, x)$, which are the values that satisfy verification.

Uniqueness. Consider a public key pk, consisting of $\vec{\mathbb{G}} = (\mathbb{G}_1, \ldots, \mathbb{G}_{n+\ell})$, $A \in \mathbb{G}_\ell$, $C \in \mathbb{G}_1$ and $\{D_{i,\beta}\}_{i\in[n],\,\beta\in\{0,1\}} \in \mathbb{G}_1^{2n}$, a value $x \in \{0,1\}^n$ and values $(y_0, \pi_0), (y_1, \pi_1) \in \mathbb{G}_{n+\ell} \times \mathbb{G}_{n+\ell-1}$ that satisfy $\mathsf{Verify}(pk, x, y_\beta, \pi_\beta) = 1$, for $\beta \in \{0,1\}$. It suffices to show that $y_0 = y_1$.

Let $\alpha, \gamma, d_{i,\beta} \in \mathbb{Z}_p$ be such that $A = g_\ell^\alpha$, $C = g^\gamma$ and $D_{i,\beta} = g^{d_{i,\beta}}$ for $i \in [n]$, $\beta \in \{0,1\}$. The first verification equation is $e(g, \pi) = e(A, D(x)) = e\big(g_\ell^\alpha, g_n^{\prod_{i\in[n]} d_{i,x_i}}\big) = (g_{n+\ell})^{\alpha \prod_{i\in[n]} d_{i,x_i}}$, which can only be satisfied by $\pi = (g_{n+\ell-1})^{\alpha \prod_{i\in[n]} d_{i,x_i}}$. We thus have $\pi_0 = \pi_1 = \pi$.

The second verification equation is $y = e(C, \pi)$. Since $e(C, \pi) = e\big(g^\gamma, (g_{n+\ell-1})^{\alpha \prod_{i\in[n]} d_{i,x_i}}\big)$, the only satisfying value for y is $y = (g_{n+\ell})^{\alpha\cdot\gamma \prod_{i\in[n]} d_{i,x_i}}$; thus $y_0 = y_1 = y$, which proves uniqueness.

Constraint-hiding. The proof algorithm P maps $sk = \big(\alpha, \gamma, \{d_{i,\beta}\}_{i\in[n],\,\beta\in\{0,1\}}\big) \in \mathbb{Z}_p^{2n+2}$ and $x \in \{0,1\}^n$ to $P(sk, x) := (g_{n+\ell-1})^{\alpha \prod_{i\in[n]} d_{i,x_i}}$. Since by provability, this is precisely the value that $\mathsf{Prove}(sk_f, x)$ outputs for any $x \in \{0,1\}^n$ and any key $sk_f \in [\mathsf{Constrain}(sk, f)]$ for any ℓ-level circuit f with $f(x) = 1$, the constraint-hiding property follows immediately.

5.3 Proof of Pseudorandomness

Theorem 2. *If there exists a PT adversary $\mathcal{A}$ that makes one challenge query and breaks pseudorandomness of the above n-bit depth-ℓ circuit-constrained VRF with advantage $\epsilon(\lambda)$ then there exists a PT algorithm $\mathcal{B}$ that breaks the $(n+\ell)$-Multilinear Decisional Diffie-Hellman assumption with advantage $2^{-n} \cdot \epsilon(\lambda)$.*

Proof. The proof follows that of [BW13] closely. Consider a PT algorithm $\mathcal{A}$ that wins the pseudorandomness game with advantage $\epsilon(\lambda)$. Without loss of generality, we assume that $\mathcal{A}$ never queries a key for a circuit f with $f(x^*) = 1$ and never queries its PROVE oracle on x^* (where x^* is the value queried to CHALLENGE). We construct an algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to break $(n+\ell)$-MDDH.

Setup. $\mathcal{B}$ receives a challenge consisting of a group sequence $\vec{\mathbb{G}}$ and values $g = g_1, g^{c_1}, \ldots, g^{c_{n+\ell+1}}$ and T, where T is either $(g_{n+\ell})^{\prod_{i\in[n+\ell+1]} c_i}$ or a random group element in $\mathbb{G}_{n+\ell}$. Using the challenge, $\mathcal{B}$ sets up the keys as follows. It chooses $x^* \leftarrow\!\!\$\ \{0,1\}^n$ and $z_1, \ldots, z_n \leftarrow\!\!\$\ \mathbb{Z}_p$ and sets

$$D_{i,\beta} := \begin{cases} g^{c_i} & \text{if } x^*_i = \beta \\ g^{z_i} & \text{if } x^*_i \neq \beta \end{cases} \qquad \text{for } i \in [n],\ \beta \in \{0,1\},$$

Repeatedly applying the bilinear maps, it computes $A := g_\ell^{c_{n+1}\cdots c_{n+\ell}}$ and sets $C := g^{c_{n+\ell+1}}$. Note that this defines $d_{i,x^*_i} = c_i$ and $d_{i,\overline{x}^*_i} = z_i$ (where $\overline{x}^*_i$ denotes $1 - x^*_i$), as well as $\alpha = c_{n+1}\cdots c_{n+\ell}$ and $\gamma = c_{n+\ell+1}$, which is distributed as in the real scheme. The parameters are set up so that we have $F(sk, x^*) = (g_{n+\ell})^{\alpha\cdot\gamma\prod_{i\in[n]} d_{i,x^*_i}} = (g_{n+\ell})^{\prod_{i\in[n+\ell+1]} c_i}$. $\mathcal{B}$ runs $\mathcal{A}$ on input $pk = (\vec{\mathbb{G}}, A, C, \{D_{i,\beta}\}_{i\in[n],\beta\in\{0,1\}})$.

Constrain queries. Suppose $\mathcal{A}$ queries a private key for a circuit f. If $f(x^*) = 1$ then $\mathcal{B}$ aborts and outputs a guess $b' \leftarrow \{0,1\}$. Otherwise, it must compute the key component K_w for every wire w of f. The simulation follows [BW13], who base their technique on [GGH$^+$13c].

For the final gate $w = n+q$ we have $r_w = \alpha$ and elements of K_{n+q} contain $(g_{\ell-1})^\alpha = (g_{\ell-1})^{c_{n+1}\cdots c_{n+\ell}}$, which $\mathcal{B}$ cannot compute. Simulating this is thus the tricky part and is done as follows. In order to compute e.g. $K_{n+q,4} = (g_{\ell-1})^{\alpha - b_{n+q}\cdot r_{B(n+q)}}$ (if the last gate is an OR gate), $\mathcal{B}$ sets $b_{n+q} := c_{n+\ell}$ and $r_{B(n+q)} := c_{n+1}\cdots c_{n+\ell-1}$ (and adds some known randomness to each), so α cancels out and $\mathcal{B}$ can compute K_{n+q}. Now $r_{B(n+q)}$ in level $\ell-1$ contains $c_{n+1}\cdots c_{n+\ell-1}$, which has one fewer challenge value. Applying the trick again, $\mathcal{B}$ chooses the randomness of $B(n+q)$'s parent gates in level $\ell-2$ as $c_{n+1}\cdots c_{n+\ell-2}$, and so on.

Note that since $f_{n+q}(x^*) = f(x^*) = 0$, if gate $n+q$ is an OR gate then both its parents must satisfy $f_{A(n+q)}(x^*) = f_{B(n+q)}(x^*) = 0$ and we need to embed challenge elements in both $r_{B(n+q)}$ and $r_{A(n+q)}$ to simulate K_{n+q}. On the other hand, for an AND gate w with $f_w(x^*) = 0$, only one of its parent gates must evaluate x^* to 0, and for the cancellation trick to work, it suffices to embed $c_{n+1}\cdots c_{n+\mathtt{depth}(w)-1}$ in the randomness of that parent.

For every gate w at level j for which $f_w(x^*) = 0$, we thus set $r_w := c_{n+1}\cdots c_{n+j}$ (plus some $\eta_w \leftarrow\!\!\$\ \mathbb{Z}_p$ to make r_w uniform). For the input wires we have $r_w := c_{n+1} + \eta_w$, for which we can simulate $K_w = g^{r_w\cdot d_{w,1}}$, since $d_{w,1} = z_w$ when $f_w(x^*) = x^*_w = 0$. Note that this does not work for wires and gates w with $f_w(x^*) = 1$, for which it however suffices to compute the key elements K_w honestly.

Formalizing the above, $\mathcal{B}$ answers a CONSTRAIN query for $f = (n, q, A, B, \mathtt{GateType})$ by computing K_w for every gate starting from the input wires:

Input wire: Suppose $w \in [n]$. If $x^*_w = 1$ then choose $r_w \leftarrow\!\!\$\ \mathbb{Z}_p$ and compute the key component

$$K_w := (D_{w,1})^{r_w} = g^{r_w\cdot d_{w,1}} \ .$$

If $x^*_w = 0$ (in which case $d_{w,1} = z_w$), we choose $\eta_w \leftarrow\!\!\$\ \mathbb{Z}_p$, implicitly set $r_w := c_{n+1} + \eta_w$ and compute

$$K_w := \left(g^{c_{n+1}} \cdot g^{\eta_w}\right)^{z_w} = g^{r_w \cdot d_{w,1}} \ .$$

OR gate: If $\texttt{GateType}(w) = \text{OR}$, we let $j = \texttt{depth}(w)$ and again distinguish two cases. If $f_w(x^*) = 1$, choose $a_w, b_w, r_w \leftarrow\!\!\$\ \mathbb{Z}_p$ and set K_w as specified by Constrain:

$$K_{w,1} := g^{a_w} \qquad K_{w,2} := g^{b_w} \qquad K_{w,3} := (g_{j-1})^{r_w - a_w \cdot r_{A(w)}} \qquad K_{w,4} := (g_{j-1})^{r_w - b_w \cdot r_{B(w)}}$$

(Even when $r_{A(w)} = c_{n+1} \cdots c_{n+j-1} + \eta_{A(w)}$, one can compute $(g_{j-1})^{r_{A(w)}}$ using the pairings.)

If $f_w(x^*) = 0$, $\mathcal{B}$ chooses $\psi_w, \phi_w, \eta_w \leftarrow\!\!\$\ \mathbb{Z}_p$ and implicitly sets $a_w := c_{n+j} + \psi_w$, $b_w := c_{n+j} + \phi_w$ and $r_w := c_{n+1} \cdots c_{n+j} + \eta_w$. Since $f_w(x^*) = 0$ implies $f_{A(w)}(x^*) = f_{B(w)}(x^*) = 0$, we have $r_{A(w)} = c_{n+1} \cdots c_{n+j-1} + \eta_{A(w)}$ and $r_{B(w)} = c_{n+1} \cdots c_{n+j-1} + \eta_{B(w)}$. This enables $\mathcal{B}$ to create the key components as follows:

$$\begin{aligned}
K_{w,1} &:= g^{c_{n+j}} \cdot g^{\psi_w} \qquad\qquad K_{w,2} := g^{c_{n+j}} \cdot g^{\phi_w} \\
K_{w,3} &:= (g_{j-1})^{\eta_w - c_{n+j} \cdot \eta_{A(w)} - \psi_w (c_{n+1} \cdots c_{n+j-1} + \eta_{A(w)})} \\
&= (g_{j-1})^{c_{n+1} \cdots c_{n+j} + \eta_w - (c_{n+j} + \psi_w) \cdot (c_{n+1} \cdots c_{n+j-1} + \eta_{A(w)})} = (g_{j-1})^{r_w - a_w \cdot r_{A(w)}} \\
K_{w,4} &:= (g_{j-1})^{\eta_w - c_{n+j} \cdot \eta_{B(w)} - \phi_w (c_{n+1} \cdots c_{n+j-1} + \eta_{B(w)})} = (g_{j-1})^{r_w - b_w \cdot r_{B(w)}}
\end{aligned}$$

(Again, $\mathcal{B}$ can compute $K_{w,3}$ and $K_{w,4}$ by computing $(g_{j-1})^{c_{n+1} \cdots c_{n+j-1}}$ via the pairings.)

AND gate: If $\texttt{GateType}(w) = \text{AND}$, we let $j = \texttt{depth}(w)$ and distinguish two cases. If $f_w(x^*) = 1$ then $\mathcal{B}$ chooses $a_w, b_w, r_w \leftarrow\!\!\$\ \mathbb{Z}_p$ and defines K_w as specified by Constrain:

$$K_{w,1} := g^{a_w} \qquad\qquad K_{w,2} := g^{b_w} \qquad\qquad K_{w,3} := (g_{j-1})^{r_w - a_w \cdot r_{A(w)} - b_w \cdot r_{B(w)}}$$

Otherwise, choose $\psi_w, \phi_w, \eta_w \leftarrow\!\!\$\ \mathbb{Z}_p$. Suppose $f_{A(w)}(x^*) = 0$. Then implicitly set $a_w := c_{n+j} + \psi_w$, $b_w := \phi_w$ and $r_w := c_{n+1} \cdots c_{n+j} + \eta_w$. Since we have $r_{A(w)} = c_{n+1} \cdots c_{n+j-1} + \eta_{A(w)}$, and since $(g_{j-1})^{r_{B(w)}}$ is computable via the pairings, $\mathcal{B}$ can compute the key components as follows:

$$\begin{aligned}
K_{w,1} &:= g^{c_{n+j}} \cdot g^{\psi_w} \qquad\qquad K_{w,2} := g^{\phi_w} \\
K_{w,3} &:= (g_{j-1})^{\eta_w - \psi_w \cdot c_{n+1} \cdots c_{n+j-1} - (c_{n+j} + \psi_w) \eta_{A(w)} - \phi_w \cdot r_{B(w)}} \\
&= (g_{j-1})^{c_{n+1} \cdots c_{n+j} + \eta_w - (c_{n+j} + \psi_w) \cdot (c_{n+1} \cdots c_{n+j-1} + \eta_{A(w)}) - \phi_w \cdot r_{B(w)}} = (g_{j-1})^{r_w - a_w \cdot r_{A(w)} - b_w \cdot r_{B(w)}}
\end{aligned}$$

If $f_{A(w)}(x^*) = 1$ then we must have $f_{B(w)}(x^*) = 0$ and $\mathcal{B}$ can compute the key components as above with the roles of a_w and b_w swapped.

Prove queries. Suppose $\mathcal{A}$ queries its PROVE oracle on x. If $x = x^*$ then $\mathcal{B}$ aborts and outputs $b' \leftarrow\!\!\$\ \{0,1\}$. Otherwise, let j be such that $x_j \neq x^*_j$. Repeatedly applying the bilinear maps, $\mathcal{B}$ computes $(g_{n-1})^{\prod_{i \in [n] \setminus \{j\}} d_{i,x_i}}$, and by raising it to $z_i = d_{i,x_i}$, obtains $H = (g_{n-1})^{\prod_{i \in [n]} d_{i,x_i}}$. This suffices to compute

$$\begin{aligned}
e(A, H) &= e\left(g_\ell^\alpha, (g_{n-1})^{\prod_{i \in [n]} d_{i,x_i}}\right) = P(sk, x) \\
e(C, P(sk, x)) &= e\left(g^\gamma, (g_{n+\ell-1})^{\alpha \prod_{i \in [n]} d_{i,x_i}}\right) = F(sk, x)
\end{aligned}$$

Challenge query. When $\mathcal{A}$ queries the challenge oracle for a value different from x^*, $\mathcal{B}$ aborts and outputs a random bit $b' \leftarrow\!\!\$\ \{0,1\}$. Otherwise it returns T, which is either $(g_{n+\ell})^{\prod_{i\in[n+\ell+1]} c_i} = (g_{n+\ell})^{\alpha\cdot\gamma\prod_{i\in[n]} d_{i,x_i^*}} = F(sk, x^*)$ or a random element from $\mathbb{G}_{n+\ell}$, thus perfectly simulating the experiment for pseudorandomness. When $\mathcal{A}$ outputs a bit b', $\mathcal{B}$ halts and returns b'.

Success probability. The probability that $\mathcal{B}$ wins the MDDH game is analyzed as for the bit-fixing VRF. Let $\texttt{abort}$ denote the event that $\mathcal{B}$ aborts during the simulation. Since $\mathcal{B}$ aborts if and only if $\mathcal{A}$ queries its CHALLENGE oracle on a value different from x^*, we have $\Pr[\texttt{abort}] = 1 - 2^{-n}$. Moreover, if $\mathcal{B}$ aborts then it outputs a random bit, thus we have $\Pr[\mathcal{B}\text{ wins} \mid \texttt{abort}] = \frac{1}{2}$. If $\mathcal{B}$ does not abort then it wins with the same probability as $\mathcal{A}$ (since $\mathcal{A}$'s success is independent of $\mathcal{B}$ guess of x^*), whose advantage is $\epsilon(\lambda)$. Thus $\Pr[\mathcal{B}\text{ wins} \mid \overline{\texttt{abort}}] = \frac{1}{2} + \epsilon(\lambda)$. Together, this yields

$$\begin{aligned}\Pr[\mathcal{B}\text{ wins}] &= \Pr[\mathcal{B}\text{ wins} \mid \texttt{abort}] \cdot \Pr[\texttt{abort}] + \Pr[\mathcal{B}\text{ wins} \mid \overline{\texttt{abort}}] \cdot \Pr[\overline{\texttt{abort}}] \\ &= \tfrac{1}{2} \cdot (1 - 2^{-n}) + \left(\tfrac{1}{2} + \epsilon(\lambda)\right) \cdot 2^{-n} \;=\; \tfrac{1}{2} + 2^{-n} \cdot \epsilon(\lambda)\ ,\end{aligned}$$

which shows that $\mathcal{B}$'s advantage in breaking $(n+\ell)$-MDDH is $2^{-n} \cdot \epsilon(\lambda)$. □

References

[ACF13] Michel Abdalla, Dario Catalano, and Dario Fiore. Verifiable random functions: Relations to identity-based key encapsulation and new constructions. *Journal of Cryptology*, pages 1–50, 2013.

[BB04a] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer, May 2004.

[BB04b] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, May 2004.

[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, August 2009.

[BF14] Mihir Bellare and Georg Fuchsbauer. Policy-based signatures. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 520–537. Springer, March 2014.

[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, March 2014.

[BGRV09] Zvika Brakerski, Shafi Goldwasser, Guy N. Rothblum, and Vinod Vaikuntanathan. Weak verifiable random functions. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 558–576. Springer, March 2009.

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 12*, pages 784–796. ACM Press, October 2012.

[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 10*, pages 131–140. ACM Press, October 2010.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.

[BS02] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324:71–90, 2002. available at `http://eprint.iacr.org/2002/080`.

[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, December 2013.

[CDNO97] Ran Canetti, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky. Deniable encryption. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 90–104. Springer, August 1997.

[CL07] Melissa Chase and Anna Lysyanskaya. Simulatable VRFs with applications to multi-theorem NIZK. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 303–322. Springer, August 2007.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 476–493. Springer, August 2013.

[Dod03] Yevgeniy Dodis. Efficient construction of (distributed) verifiable random functions. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 1–17. Springer, January 2003.

[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, January 2005.

[FKPR14] Georg Fuchsbauer, Momchil Konstantinov, Krzysztof Pietrzak, and Vanishree Rao. Adaptive security of constrained prfs. Cryptology ePrint Archive, Report 2014/416, 2014. `http://eprint.iacr.org/`.

[FS12] Dario Fiore and Dominique Schröder. Uniqueness is a different story: Impossibility of verifiable random functions from trapdoor permutations. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 636–653. Springer, March 2012.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 1–17. Springer, May 2013.

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

[GGH+13c] Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 479–499. Springer, August 2013.

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

[HSW13a] Susan Hohenberger, Amit Sahai, and Brent Waters. Full domain hash from (leveled) multilinear maps and identity-based aggregate signatures. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 494–512. Springer, August 2013.

[HSW13b] Susan Hohenberger, Amit Sahai, and Brent Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. Cryptology ePrint Archive, Report 2013/509, 2013. `http://eprint.iacr.org/2013/509`.

[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, May 2010.

[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 13*, pages 669–684. ACM Press, November 2013.

[Lis05] Moses Liskov. Updatable zero-knowledge databases. In Bimal K. Roy, editor, *ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 174–198. Springer, December 2005.

[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, August 2002.

[MR01] Silvio Micali and Leonid Reyzin. Soundness in the public-key model. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 542–565. Springer, August 2001.

[MR02] Silvio Micali and Ronald L. Rivest. Micropayments revisited. In Bart Preneel, editor, *CT-RSA 2002*, volume 2271 of *LNCS*, pages 149–163. Springer, February 2002.

[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.

[NR97] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th FOCS*, pages 458–467. IEEE Computer Society Press, October 1997.

[SW13] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. Cryptology ePrint Archive, Report 2013/454, 2013. `http://eprint.iacr.org/2013/454`.

A Constrained PRFs With Fewer Group Levels

In this appendix we show that the two constrained PRFs based on multilinear groups by Boneh and Waters [BW13] can be defined with fewer levels in the group hierarchy. Constrained PRFs are basically defined like constrained VRFs but without public keys and without the Verify algorithm. Instead of Prove, there is an algorithm Eval which takes a constrained key and an input x and outputs $F(k,x)$. Besides correctness, it is required that a constrained PRF satisfies pseudorandomness, which is defined as for constrained VRFs, but with the PROVE oracle replaced by a EVAL oracle, which queried on x outputs $F(k,x)$. For the precise definition of constrained PRFs we refer to [BW13].

A.1 Bit-Fixing PRF

We show that the bit-fixing PRF for inputs of length n from [BW13] can be defined over a sequence of groups $\mathbb{G}_1,\ldots,\mathbb{G}_\kappa$ with $\kappa := n-1$ rather than $\kappa = n+1$, as in [BW13]. There the function value is defined as $F_{\mathrm{BW}}(k,x) := (g_{n+1})^{\alpha\prod_{i\in[n]} d_{i,x_i}}$, which we show can be replaced by $F(k,x) := (g_{n-1})^{\prod_{i\in[n]} d_{i,x_i}}$, while removing α from the secret key. We adapt the definition of constrained keys and prove the construction pseudorandom under the $(n-1)$-MDDH assumption.

$\underline{\mathsf{Setup}(1^\lambda,1^n)}$: On input the security parameter λ and the input length n, the setup runs $\mathcal{G}(1^\lambda,n-1)$, which outputs a sequence of groups $\vec{\mathbb{G}} = (\mathbb{G}_1,\ldots,\mathbb{G}_{n-1})$ of prime order p, with generators $g = g_1,\ldots,g_{n-1}$. It chooses $(d_{1,0},d_{1,1}),\ldots,(d_{n,0},d_{n,1}) \leftarrow\!\!\$\ \mathbb{Z}_p^2$ and sets $D_{i,\beta} := g^{d_{i,\beta}}$ for $i\in[n]$ and $\beta\in\{0,1\}$. The PRF master key is $k := \big(\vec{\mathbb{G}} = (\mathbb{G}_1,\ldots,\mathbb{G}_n), \{d_{i,\beta}\}_{i\in[n],\,\beta\in\{0,1\}}, \{D_{i,\beta}\}_{i\in[n],\,\beta\in\{0,1\}}\big)$.

The domain and range are defined as $\mathcal{X} := \{0,1\}^n$ and $\mathcal{Y} := \mathbb{G}_{n-1}$. The keyed function F for input $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ is defined as

$$F(k,x) := (g_{n-1})^{\prod_{i\in[n]} d_{i,x_i}} \ .$$

$\underline{\mathsf{Constrain}(k, \mathbf{v})}$: The algorithm takes as input k and a vector $\mathbf{v} \in \{0,1,?\}^n$ describing the constrained input space as $S_{\mathbf{v}} := \{x \in \{0,1\}^n \,|\, \forall i \in [n] : x_i = \mathbf{v}_i \vee \mathbf{v}_i = ?\}$. Let $V := \{i \in [n] \,|\, \mathbf{v}_i \neq ?\}$ be the set of indices fixed by $\mathbf{v}$. Return $k_{\mathbf{v}} = (\vec{\mathbb{G}}, \{D_{i,\beta}\}_{i\in[n]\setminus V, \beta\in\{0,1\}}, k'_{\mathbf{v}})$, with $k'_{\mathbf{v}}$ defined as follows:

If $|V| > 1$ then set $k'_{\mathbf{v}} := (g_{|V|-1})^{\prod_{i\in V} d_{i,\mathbf{v}_i}}$; if $V = \{j\}$ then set $k'_{\mathbf{v}} := d_{j,\mathbf{v}_j}$.

$\underline{\mathsf{Eval}(k_{\mathbf{v}}, x)}$: Let V be as above and let $\overline{V} := \{i \in [n] \,|\, \mathbf{v}_i = ?\}$ be its complement; if $x_i \neq \mathbf{v}_i$ for some $i \in V$ then abort. Using the bilinear maps applied to $\{D_{i,x_i}\}_{i\in\overline{V}}$, first compute

$$D_{\overline{V}}(x) = (g_{|\overline{V}|})^{\prod_{i\in\overline{V}} d_{i,x_i}} \ .$$

If $|V| > 1$, output $e(D_{\overline{V}}(x), k'_{\mathbf{v}}) = e\big((g_{|\overline{V}|})^{\prod_{i\in\overline{V}} d_{i,x_i}}, (g_{|V|-1})^{\prod_{i\in V} d_{i,\mathbf{v}_i}}\big) = (g_{n-1})^{\prod_{i\in[n]} d_{i,x_i}} = F(k,x)$.
If $V = \{j\}$, output $D_{\overline{V}}(x)^{k'_{\mathbf{v}}} = \big((g_{|\overline{V}|})^{\prod_{i\in\overline{V}} d_{i,x_i}}\big)^{d_{j,\mathbf{v}_j}} = (g_{n-1})^{\prod_{i\in[n]} d_{i,x_i}} = F(k,x)$.

The above construction is pseudorandom:

Theorem 3. *If there exists a PT adversary $\mathcal{A}$ that breaks pseudorandomness of the above n-bit input bit-fixing PRF with advantage $\epsilon(\lambda)$ then there exists a PT algorithm $\mathcal{B}$ that breaks the $(n-1)$-Multilinear Decisional Diffie-Hellman assumption with advantage $2^{-n} \cdot \epsilon(\lambda)$.*

Proof. We assume that when x^* is $\mathcal{A}$'s (one-time) challenge query then $\mathcal{A}$ never queries constrained keys that could evaluate x^*, nor its EVAL oracle on x^*. We construct $\mathcal{B}$, which receives an $(n-1)$-MDDH challenge consisting of a group-sequence description $\vec{\mathbb{G}}$ and elements $g = g_1, g^{c_1}, \ldots, g^{c_n}$ and T, which is either $(g_{n-1})^{\prod_{i\in[n]} c_j}$ or a random element from $\mathbb{G}_{n-1}$. $\mathcal{B}$ picks $x^* \leftarrow\!\!\$\ \{0,1\}^n$, which it hopes will be $\mathcal{A}$'s challenge query, and $z_1, \ldots, z_n \leftarrow\!\!\$\ \mathbb{Z}_p$ and sets

$$D_{i,\beta} := \begin{cases} g^{c_i} & \text{if } x^*_i = \beta \\ g^{z_i} & \text{if } x^*_i \neq \beta \end{cases}$$

for $i \in [n]$ and $\beta \in \{0,1\}$. Observe that this is distributed as in the real scheme.

Constrain queries. Suppose $\mathcal{A}$ queries a secret key for $\mathbf{v} \in \{0,1,?\}^n$. Let $V := \{i \in [n] \,|\, \mathbf{v}_i \neq ?\}$ be the set of fixed indices. $\mathcal{B}$ chooses an arbitrary $j \in V$ such that $\mathbf{v}_j \neq x^*_j$. If no such j exists then the key could be used to evaluate $F(k, x^*)$, in which case $\mathcal{B}$ aborts outputting a random guess $b' \leftarrow\!\!\$\ \{0,1\}$.

If $|V| = 1$ then $V = \{j\}$, thus $\mathcal{B}$ knows z_j with $D_{j,\mathbf{v}_j} = g^{z_j}$ and sets $k'_{\mathbf{v}} := z_j$. If $|V| > 1$, then by repeatedly applying the bilinear maps to the values $\{D_{i,\mathbf{v}_i}\}_{i\in V\setminus\{j\}}$, it computes $(g_{|V|-1})^{\prod_{i\in V\setminus\{j\}} d_{i,\mathbf{v}_i}}$ and raises this value to $z_j = d_{j,\mathbf{v}_j}$ to compute $k'_{\mathbf{v}} := (g_{|V|-1})^{\prod_{i\in V} d_{i,\mathbf{v}_i}}$. $\mathcal{B}$ answers the query with $k_{\mathbf{v}} := (\vec{\mathbb{G}}, \{D_{i,\beta}\}_{i\in[n]\setminus V, \beta\in\{0,1\}}, k'_{\mathbf{v}})$.

Evaluate queries. Since $F(k,x)$ is identical to a key for $\mathbf{v} = x$, this value can be computed as for the constrained-key query above.

Challenge query. If $\mathcal{A}$'s (one-time) challenge query is different from x^* then $\mathcal{B}$ outputs a guess $b' \leftarrow\!\!\$\ \{0,1\}$. Otherwise, replies with T. If $T = (g_{n-1})^{\prod_{i\in[n]} c_j}$ then $T = (g_{n-1})^{\prod_{i\in[n]} d_{i,x^*_i}} = F(k,x^*)$. When $\mathcal{A}$ outputs a guess b' then $\mathcal{B}$, if it has not aborted, outputs the same guess b'.

Success probability. The probability that $\mathcal{B}$ wins the MDDH game is analyzed as for the bit-fixing VRF. Let $\texttt{abort}$ denote the event that $\mathcal{B}$ aborts during the simulation. Since $\mathcal{B}$ aborts if and only if $\mathcal{A}$ queries its CHALLENGE oracle on a value different from x^*, we have $\Pr[\texttt{abort}] = 1 - 2^{-n}$. Moreover, if $\mathcal{B}$ aborts then it outputs a random bit, thus we have $\Pr[\mathcal{B} \text{ wins} \,|\, \texttt{abort}] = \frac{1}{2}$. If $\mathcal{B}$ does not abort then it wins with the same probability as $\mathcal{A}$ (since $\mathcal{A}$'s success is independent of $\mathcal{B}$ guess of x^*), whose advantage is $\epsilon(\lambda)$. Thus $\Pr[\mathcal{B} \text{ wins} \,|\, \overline{\texttt{abort}}] = \frac{1}{2} + \epsilon(\lambda)$. Together, this yields

$$\begin{aligned}\Pr[\mathcal{B} \text{ wins}] &= \Pr[\mathcal{B} \text{ wins} \,|\, \texttt{abort}] \cdot \Pr[\texttt{abort}] + \Pr[\mathcal{B} \text{ wins} \,|\, \overline{\texttt{abort}}] \cdot \Pr[\overline{\texttt{abort}}] \\ &= \tfrac{1}{2} \cdot (1 - 2^{-n}) + \left(\tfrac{1}{2} + \epsilon(\lambda)\right) \cdot 2^{-n} \;=\; \tfrac{1}{2} + 2^{-n} \cdot \epsilon(\lambda) \;\;,\end{aligned}$$

which shows that $\mathcal{B}$'s advantage in breaking MDDH is $2^{-n} \cdot \epsilon(\lambda)$. □

A.2 Constrained PRF for Circuit Predicates

The construction is almost identical to the one by Boneh and Waters [BW13], except that we set $\kappa := n + \ell - 1$ rather than $\kappa := n + \ell$, where n is the input length and ℓ the maximum depth of the circuits. In the Constrain algorithm, we then define the key elements for input wires as elements from $\mathbb{G}_1$ rather than $\mathbb{G}_2$ and key elements for gates of depth j will contain values from $\mathbb{G}_{j-1}$ rather than $\mathbb{G}_j$. Moreover, the values E_w computed by Eval for a gate w of depth j will be in $\mathbb{G}_{j+n-1}$ rather than $\mathbb{G}_{j+n}$; in particular the value of the output gate (which is $F(k,x)$) is in $\mathbb{G}_{n+\ell-1}$ instead of $\mathbb{G}_{n+\ell}$.

Setup$(1^\lambda, 1^n, 1^\ell)$: On input the security parameter, the bit length n and the maximum circuit depth ℓ do the following: Run $\mathcal{G}(1^\lambda, \kappa)$ with $\kappa := n + \ell - 1$ to compute a sequence of groups $\vec{\mathbb{G}} = (\mathbb{G}_1, \ldots, \mathbb{G}_\kappa)$ of prime order p, with generators $g_1, \ldots, g_\kappa$, and let $g = g_1$. Choose random exponents $\alpha \leftarrow\!\!\$\, \mathbb{Z}_p$ and $(d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}) \leftarrow\!\!\$\, \mathbb{Z}_p^2$ and set $D_{i,\beta} := g^{d_{i,\beta}}$ for $i \in [n]$ and $\beta \in \{0,1\}$. The key is defined as $k := \big(\vec{\mathbb{G}} = (\mathbb{G}_1, \ldots, \mathbb{G}_\kappa), \alpha, \{d_{i,\beta}\}_{i\in[n],\,\beta\in\{0,1\}}, \{D_{i,\beta}\}_{i\in[n],\,\beta\in\{0,1\}}\big)$.

The domain is $\mathcal{X} = \{0,1\}^n$ and the range is $\mathcal{Y} = \mathbb{G}_\kappa$. On input $x = (x_1, \ldots, x_n) \in \mathcal{X}$, the function value is defined as

$$F(k,x) = g_\kappa^{\alpha \prod_{i\in[n]} d_{i,x_i}} \;\;.$$

Constrain$(k, f = (n, q, A, B, \texttt{GateType}))$: On input the key and a circuit description f with n input wires, q gates (and thus $n + q$ wires), with the wire $n + q$ designated as output wire, Constrain first chooses $r_1, \ldots, r_{n+q-1} \leftarrow\!\!\$\, \mathbb{Z}_p$ and sets $r_{n+q} := \alpha$. For every wire w, generate a key component K_w, whose structure depends on the type of the wire: input wire, OR gate, or AND gate.

Input wire: If $w \in [n]$ then it corresponds to the w-th input and the key component is:

$$K_w := g^{r_w \cdot d_{w,1}} \;\;.$$

OR gate: If $w \in \mathsf{Gates}$ with $\texttt{GateType}(w) = \text{OR}$ and depth $j = \texttt{depth}(w)$ then choose $a_w, b_w \leftarrow\!\!\$\, \mathbb{Z}_p$ and compute K_w as:

$$K_{w,1} := g^{a_w} \qquad K_{w,2} := g^{b_w} \qquad K_{w,3} := (g_{j-1})^{r_w - a_w \cdot r_{A(w)}} \qquad K_{w,4} := (g_{j-1})^{r_w - b_w \cdot r_{B(w)}}$$

AND gate: If $w \in \mathsf{Gates}$ with $\texttt{GateType}(w) = \text{AND}$ and $\texttt{depth}(w) = j$ then choose $a_w, b_w \leftarrow\!\!\$\, \mathbb{Z}_p$ and compute K_w as:

$$K_{w,1} := g^{a_w} \qquad K_{w,2} := g^{b_w} \qquad K_{w,3} := (g_{j-1})^{r_w - a_w \cdot r_{A(w)} - b_w \cdot r_{B(w)}}$$

The constrained key k_f consists of components K_w for all $n+q$ wires together with the circuit description f and the elements $\{D_{i,\beta}\}_{i\in[n],\,\beta\in\{0,1\}}$.

$\underline{\mathsf{Eval}(k_f, x)}$: Given a constrained key k_f for circuit $f = (n, q, A, B, \mathtt{GateType})$ and input $x \in \{0,1\}^n$, abort if $f(x) = 0$. Otherwise, evaluate the circuit level by level starting from the input wires. For every wire w that evaluates to 1, compute the value $E_w := (g_{n+j-1})^{r_w \prod_i d_{i,x_i}}$, where $j = \mathtt{depth}(w)$. Note that since $r_{n+q} = \alpha$, we have $E_{n+q} = F(k, x)$. For every wire we distinguish the following cases:

Input wire: If $w \in [n]$, we only consider those w for which $x_w = f_w(x) = 1$. By repeatedly applying the multilinear maps to the values $\{D_{i,x_i}\}_{i\neq w}$, compute $D_{-w}(x) := (g_{n-1})^{\prod_{i\neq w} d_{i,x_i}}$ and pair it with $K_w = g^{r_w \cdot d_{w,1}}$ to obtain $E_w := {g_n}^{r_w \prod_i d_{i,x_i}}$.

OR gate: Let $w \in \mathsf{Gates}$ be such that $f_w(x) = 1$ and $\mathtt{GateType}(w) = \text{OR}$ and let $j = \mathtt{depth}(w)$. Apply the bilinear maps to $\{D_{i,x_i}\}_{i\in[n]}$ to compute $D(x) := {g_n}^{\prod_i d_{i,x_i}}$. If $f_{A(w)}(x) = 1$ then compute:

$$\begin{aligned} &e(E_{A(w)}, K_{w,1}) \cdot e(K_{w,3}, D(x)) \\ &= e\big((g_{n+j-2})^{r_{A(w)} \prod_i d_{i,x_i}}, g^{a_w}\big) \cdot e\big((g_{j-1})^{r_w - a_w \cdot r_{A(w)}}, {g_n}^{\prod_i d_{i,x_i}}\big) = (g_{j+n-1})^{r_w \prod_i d_{i,x_i}} = E_w \end{aligned}$$

Otherwise, we must have $f_{B(w)}(x) = 1$, so compute:

$$e(E_{B(w)}, K_{w,2}) \cdot e(K_{w,4}, D(x)) = (g_{j+n-1})^{r_w \prod_i d_{i,x_i}} = E_w$$

AND gate: Let $w \in \mathsf{Gates}$ be such that $f_w(x) = 1$, $\mathtt{GateType}(w) = \text{AND}$ and let $j = \mathtt{depth}(w)$. We have $f_{A(w)} = f_{B(w)} = 1$ and with $D(x)$ as above we compute:

$$\begin{aligned} &e(E_{A(w)}, K_{w,1}) \cdot e(E_{B(w)}, K_{w,2}) \cdot e(K_{w,3}, D(x)) \\ &= e\big((g_{n+j-2})^{r_{A(w)} \prod_i d_{i,x_i}}, g^{a_w}\big) \cdot e\big((g_{n+j-2})^{r_{B(w)} \prod_i d_{i,x_i}}, g^{b_w}\big) \cdot e\big((g_{j-1})^{r_w - a_w \cdot r_{A(w)} - b_w \cdot r_{B(w)}}, {g_n}^{\prod_i d_{i,x_i}}\big) \\ &= (g_{j+n-1})^{r_w \prod_i d_{i,x_i}} = E_w \end{aligned}$$

Evaluating all wires w in order, we arrive at $E_{n+q} = (g_{n+\ell-1})^{\alpha \prod_i d_{i,x_i}} = F(k, x)$.

Pseudorandomness. The above construction satisfies pseudorandomness for constrained PRFs:

Theorem 4. *If there exists a PT adversary $\mathcal{A}$ that makes 1 challenge query and breaks pseudorandomness of the above n-bit depth-ℓ circuit-constrained PRF with advantage $\epsilon(\lambda)$ then there exists a PT algorithm $\mathcal{B}$ that breaks the $(n+\ell-1)$-Multilinear Decisional Diffie-Hellman assumption with advantage $2^{-n} \cdot \epsilon(\lambda)$.*

Proof. Consider a PT algorithm $\mathcal{A}$ that wins the pseudorandomness game with advantage $\epsilon(\lambda)$. We assume that $\mathcal{A}$ only queries its CHALLENGE oracle once, say on x^*. Without loss of generality, we moreover assume that $\mathcal{A}$ never queries its CONSTRAIN oracle on a circuit f with $f(x^*) = 1$ and it never queries its EVAL oracle on x^*. We construct an algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to break $(n+\ell-1)$-MDDH.

Setup. $\mathcal{B}$ receives a challenge consisting of a group sequence $\vec{\mathbb{G}}$ and $g = g_1, g^{c_1}, \ldots, g^{c_{n+\ell}}$ along with T, where T is either $(g_{n+\ell-1})^{\prod_{i\in[n+\ell]} c_i}$ or a random element from $\mathbb{G}_{n+\ell-1}$. Using the challenge, $\mathcal{B}$ implicitly defines the key k as follows: It chooses $x^* \leftarrow\!\!\$\ \{0,1\}^n$ and $z_1, \ldots, z_n \leftarrow\!\!\$\ \mathbb{Z}_p$ and sets

$$D_{i,\beta} := \begin{cases} g^{c_i} & \text{if } x^*_i = \beta \\ g^{z_i} & \text{if } x^*_i \neq \beta \end{cases}$$

for $i \in [n]$, $\beta \in \{0,1\}$. This defines $d_{i,x_i^*} := c_i$ and $d_{i,\overline{x}_i^*} := z_i$ (where $\overline{x}_i^*$ denotes $1 - x_i^*$); moreover we will view α as $c_{n+1} \cdots c_{n+\ell}$. All values are distributed as in the real scheme.

Constrain queries. Suppose that $\mathcal{A}$ queries a constrained key for circuit $f = (n, q, A, B, \texttt{GateType})$. If $f(x^*) = 1$ then $\mathcal{B}$ aborts and outputs a random guess $b' \leftarrow\!\!\$\ \{0,1\}$. Otherwise, it produces the key component K_w for every wire w of f starting from the input wires:

Input wire: Suppose $w \in [n]$. If $x_w^* = 1$ then choose $r_w \leftarrow\!\!\$\ \mathbb{Z}_p$ and compute the key component:

$$K_w = (D_{w,1})^{r_w} := g^{r_w \cdot d_{w,1}} \quad .$$

If $x_w^* = 0$ (in which case $d_{w,1} = z_w$), choose $\eta_w \leftarrow\!\!\$\ \mathbb{Z}_p$, implicitly set $r_w := c_{n+1} + \eta_w$ and compute

$$K_w := \left(g^{c_{n+1}} \cdot g^{\eta_w}\right)^{z_w} = g^{r_w \cdot d_{w,1}} \quad .$$

OR gate: For $w \in \mathsf{Gates}$ with $\texttt{GateType}(w) = \text{OR}$ and $\texttt{depth}(w) = j$, again distinguish two cases. If $f_w(x^*) = 1$, choose $a_w, b_w, r_w \leftarrow\!\!\$\ \mathbb{Z}_p$ and create the components as specified by $\mathsf{Constrain}$:

$$K_{w,1} := g^{a_w} \qquad K_{w,2} := g^{b_w} \qquad K_{w,3} := (g_{j-1})^{r_w - a_w \cdot r_{A(w)}} \qquad K_{w,4} := (g_{j-1})^{r_w - b_w \cdot r_{B(w)}}$$

(If e.g. $r_{A(w)}$ contains $c_{n+1} \cdots c_{n+j-1}$, compute $(g_{j-1})^{r_{A(w)}}$ via the pairings.) If $f_w(x^*) = 0$ then $\mathcal{B}$ chooses $\psi_w, \phi_w, \eta_w \leftarrow\!\!\$\ \mathbb{Z}_p$ and implicitly sets $a_w := c_{n+j} + \psi_w$, $b_w := c_{n+j} + \phi_w$ and $r_w := c_{n+1} \cdots c_{n+j} + \eta_w$. Since $f_{A(w)}(x^*) = f_{B(w)}(x^*) = 0$, we have $r_{A(w)} := c_{n+1} \cdots c_{n+j-1} + \eta_{A(w)}$ and $r_{B(w)} := c_{n+1} \cdots c_{n+j-1} + \eta_{B(w)}$. This enables $\mathcal{B}$ to create the key components as follows:

$$\begin{aligned}
K_{w,1} &:= g^{c_{n+j}} \cdot g^{\psi_w} \qquad K_{w,2} := g^{c_{n+j}} \cdot g^{\phi_w} \\
K_{w,3} &:= (g_{j-1})^{\eta_w - c_{n+j} \cdot \eta_{A(w)} - \psi_w (c_{n+1} \cdots c_{n+j-1} + \eta_{A(w)})} \\
&= (g_{j-1})^{c_{n+1} \cdots c_{n+j} + \eta_w - (c_{n+j} + \psi_w) \cdot (c_{n+1} \cdots c_{n+j-1} + \eta_{A(w)})} = (g_{j-1})^{r_w - a_w \cdot r_{A(w)}} \\
K_{w,4} &:= (g_{j-1})^{\eta_w - c_{n+j} \cdot \eta_{B(w)} - \phi_w (c_{n+1} \cdots c_{n+j-1} + \eta_{B(w)})} = (g_{j-1})^{r_w - b_w \cdot r_{B(w)}}
\end{aligned}$$

AND gate: For $w \in \mathsf{Gates}$ with $\texttt{GateType}(w) = \text{AND}$ and $\texttt{depth}(w) = j$ again distinguish two cases. If $f_w(x^*) = 1$, choose $a_w, b_w, r_w \leftarrow\!\!\$\ \mathbb{Z}_p$ and compute:

$$K_{w,1} := g^{a_w} \qquad K_{w,2} := g^{b_w} \qquad K_{w,3} := (g_{j-1})^{r_w - a_w \cdot r_{A(w)} - b_w \cdot r_{B(w)}}$$

If $f_w(x^*) = 0$ then choose $\psi_w, \phi_w, \eta_w \leftarrow\!\!\$\ \mathbb{Z}_p$. If $f_{A(w)}(x^*) = 0$ then implicitly set $a_w := c_{n+j} + \psi_w$, $b_w := \phi_w$, $r_w := c_{n+1} \cdots c_{n+j} + \eta_w$. Since we have $r_{A(w)} = c_{n+1} \cdots c_{n+j-1} + \eta_{A(w)}$ and since $(g_{j-1})^{r_{B(w)}}$ can be computed via the pairings, $\mathcal{B}$ can compute the key components as follows

$$\begin{aligned}
K_{w,1} &:= g^{c_{n+j}} \cdot g^{\psi_w} \qquad K_{w,2} := g^{\phi_w} \\
K_{w,3} &:= (g_{j-1})^{\eta_w - \psi_w \cdot c_{n+1} \cdots c_{n+j-1} - (c_{n+j} + \psi_w) \eta_{A(w)} - \phi_w \cdot r_{B(w)}} \\
&= (g_{j-1})^{c_{n+1} \cdots c_{n+j} + \eta_w - (c_{n+j} + \psi_w) \cdot (c_{n+1} \cdots c_{n+j-1} + \eta_{A(w)}) - \phi_w \cdot r_{B(w)}} = (g_{j-1})^{r_w - a_w \cdot r_{A(w)} - b_w \cdot r_{B(w)}}
\end{aligned}$$

If $f_{A(w)}(x^*) = 1$ then we must have $f_{B(w)}(x^*) = 0$ and $\mathcal{B}$ can compute the key components as above with the roles of a_w and b_w reversed.

Evaluate queries. Suppose $\mathcal{A}$ queries EVAL on x. If $x = x^*$ then $\mathcal{B}$ aborts and outputs $b' \leftarrow\!\!\$\ \{0,1\}$. Otherwise, let j be such that $x_j \neq x_j^*$. Repeatedly applying the pairings, $\mathcal{B}$ computes $g_\ell^{c_{n+1}\cdots c_{n+\ell}} = g_\ell^\alpha =: A$, as well as $(g_{n-1})^{\prod_{i\in[n]\setminus\{j\}} d_{i,x_i}}$, and by raising it to $z_i = d_{i,x_i}$, obtains $H := (g_{n-1})^{\prod_{i\in[n]} d_{i,x_i}}$. From this it computes

$$e(A,H) = e\big(g_\ell^\alpha, (g_{n-1})^{\prod_{i\in[n]} d_{i,x_i}}\big) = F(k,x) \ .$$

Challenge query. When $\mathcal{A}$ queries the challenge oracle for a value different from x^*, $\mathcal{B}$ aborts and outputs $b' \leftarrow\!\!\$\ \{0,1\}$. Otherwise, it returns T, which is either $(g_{n+\ell-1})^{\prod_{i\in[n+\ell]} c_i} = (g_{n+\ell-1})^{\alpha\prod_{i\in[n]} d_{i,x_i^*}} = F(k,x^*)$ or a random group element in $\mathbb{G}_{n+\ell-1}$, thus perfectly simulating the pseudorandomness experiment. When $\mathcal{A}$ terminates outputting a bit b', $\mathcal{B}$ halts and returns b'.

Success probability. The probability that $\mathcal{B}$ wins the MDDH game is analyzed as for the bit-fixing VRF. Let `abort` denote the event that $\mathcal{B}$ aborts during the simulation. Since $\mathcal{B}$ aborts if and only if $\mathcal{A}$ queries its CHALLENGE oracle on a value different from x^*, we have $\Pr[\texttt{abort}] = 1 - 2^{-n}$. Moreover, if $\mathcal{B}$ aborts then it outputs a random bit, thus we have $\Pr[\mathcal{B}\text{ wins} \mid \texttt{abort}] = \frac{1}{2}$. If $\mathcal{B}$ does not abort then it wins with the same probability as $\mathcal{A}$ (since $\mathcal{A}$'s success is independent of $\mathcal{B}$ guess of x^*), whose advantage is $\epsilon(\lambda)$. Thus $\Pr[\mathcal{B}\text{ wins} \mid \overline{\texttt{abort}}] = \frac{1}{2} + \epsilon(\lambda)$. Together, this yields

$$\begin{aligned}\Pr[\mathcal{B}\text{ wins}] &= \Pr[\mathcal{B}\text{ wins} \mid \texttt{abort}]\cdot\Pr[\texttt{abort}] + \Pr[\mathcal{B}\text{ wins} \mid \overline{\texttt{abort}}]\cdot\Pr[\overline{\texttt{abort}}] \\ &= \tfrac{1}{2}\cdot(1-2^{-n}) + \big(\tfrac{1}{2}+\epsilon(\lambda)\big)\cdot 2^{-n} \ = \ \tfrac{1}{2} + 2^{-n}\cdot\epsilon(\lambda) \ ,\end{aligned}$$

which shows that $\mathcal{B}$'s advantage in breaking MDDH is $2^{-n}\cdot\epsilon(\lambda)$. □

B Implementing Our Constructions With Graded Encodings

B.1 Graded Encoding Systems.

Garg, Gentry and Halevi [GGH13a] define an "approximate" version of multilinear groups, which they call *graded encoding systems*. Roughly speaking, to an element of a multilinear group, such as g_i^α corresponds a *set* $S_i^{(\alpha)}$ of bit strings in a graded encoding system. These encodings permit additive homomorphisms (corresponding to the group operation in $\mathbb{G}_i$) and a bounded multiplicative homomorphism (corresponding to the multilinear map e).

Formally, a κ-graded encoding system consists of a ring R and a system of sets $\mathcal{S} = \{S_i^{(\alpha)} \subset \{0,1\}^* \mid i \in [\kappa], \alpha \in R\}$, such that for every $i \in [\kappa]$ and every distinct $\alpha_1, \alpha_2 \in R$, the sets $S_i^{(\alpha_1)}$ and $S_i^{(\alpha_2)}$ are disjoint. For $\alpha \in R$, the set $S_i^{(\alpha)}$ represents the possible level-i encodings of α, and we define $S_i := \bigcup_{\alpha\in R} S_i^{(\alpha)}$, which corresponds to the group $\mathbb{G}_i$. We require that there exist the following efficient procedures:

Instance generation: On input the security parameter and the number of levels κ, $\mathsf{InstGen}(1^\lambda, 1^\kappa)$ outputs parameters *params*, which describe a κ-graded encoding system and $\mathbf{p}_{\mathrm{zt}}$, the *zero-test element* for level κ. The value *params* will be an implicit input to all of the following procedures.

Ring sampler: $\mathsf{samp}()$ outputs a *level-0 encoding* $a \in \mathcal{S}_0^{(\alpha)}$ for a uniformly random $\alpha \leftarrow\!\!\$\ R$.

Encoding: $\mathsf{enc}(i,a)$ takes a level $i \in [\kappa]$ and a level-0 encoding $a \in S_0^{(\alpha)}$ for some $\alpha \in R$, and outputs a *level-i* encoding $u \in S_i^{(\alpha)}$ for the same α.

Re-randomization: $\mathsf{reRand}(u)$ re-randomizes a level-i encoding u of α, outputting an element of $S_i^{(\alpha)}$. Given two encodings $u_0, u_1 \in S_i^{(\alpha)}$, the output distributions of $u_0' \leftarrow\!\!\$\ \mathsf{reRand}(i, u_0)$ and $u_1' \leftarrow\!\!\$\ \mathsf{reRand}(i, u_1)$ are statistically the same.[2]

Addition and negation: Given $i \in [\kappa]$ and two encodings $u_1 \in S_i^{(\alpha_1)}$ and $u_2 \in S_i^{(\alpha_2)}$, $\mathsf{add}(i, u_1, u_2)$ outputs an encoding $u \in S_i^{(\alpha_1+\alpha_2)}$ and $\mathsf{neg}(i, u_1)$ outputs an encoding $u \in S_i^{(-\alpha_1)}$.

Multiplication: For $u_1 \in S_i^{(\alpha_1)}$ and $u_2 \in S_j^{(\alpha_2)}$, with $i + j \le \kappa$, $\mathsf{mul}(i, u_1, j, u_2)$ outputs an encoding $u \in S_{i+j}^{(\alpha_1 \cdot \alpha_2)}$.

Zero-test: Algorithm $\mathsf{isZero}(\mathbf{p}_{\mathrm{zt}}, u)$ outputs 1 if $u \in S_\kappa^{(0)}$ and outputs 0 otherwise. Together with negation and addition, this yields an equality test for elements in S_κ.

Extraction: This outputs a *canonical, random* representation of ring elements from one of their level-κ encodings: $\mathsf{ext}(\mathbf{p}_{\mathrm{zt}}, u)$ outputs $K \in \{0,1\}^{\ell(\lambda)}$, such that for any $\alpha \in R$, and any $u_1, u_2 \in S_\kappa^{(\alpha)}$, we have

$$\mathsf{ext}(\mathbf{p}_{\mathrm{zt}}, u_1) = \mathsf{ext}(\mathbf{p}_{\mathrm{zt}}, u_2)$$

with overwhelming probability. Moreover, for a uniform $\alpha \leftarrow\!\!\$\ R$ and any $u \in S_\kappa^{(\alpha)}$, the output of $\mathsf{ext}(\mathbf{p}_{\mathrm{zt}}, u)$ is statistically uniform in $\{0,1\}^{\ell(\lambda)}$.

The GDDH assumption. Gentry et al. [GGH13a] define a hardness assumption for graded encodings, which corresponds to the κ-multilinear decisional Diffie-Hellman assumption in multilinear groups. Consider the following game:

- The challenger runs $(\mathit{params}, \mathbf{p}_{\mathrm{zt}}) \leftarrow\!\!\$\ \mathsf{InstGen}(1^\lambda, 1^\kappa)$, and for $i = 1, \ldots, \kappa + 1$, samples level-0 encodings of random elements $\alpha_i \leftarrow\!\!\$\ R$ and computes level-1 encodings of them:

$$a_i \leftarrow\!\!\$\ \mathsf{samp}() \qquad\qquad u_i \leftarrow\!\!\$\ \mathsf{enc}(1, a_i)$$

- It computes a level-0 encoding of $\prod_{i\in[\kappa+1]} \alpha_i$ by setting $\tilde{a} \leftarrow\!\!\$\ \prod_{i\in[\kappa+1]} a_i$, where for $u_1, \ldots, u_n \in S_0$ we use the product notation to denote

$$\textstyle\prod_{i\in[n]} u_i := \mathsf{mul}\big(0, \ldots \mathsf{mul}\big(0, \mathsf{mul}(0, u_1, 0, u_2), 0, u_3\big), \ldots, 0, u_n\big) \ . \tag{2}$$

- It chooses a level-0 encoding of a random element: $\hat{a} \leftarrow\!\!\$\ \mathsf{samp}()$. It flips a bit $b \leftarrow\!\!\$\ \{0,1\}$, sets $T_0 \leftarrow\!\!\$\ \mathsf{enc}(\kappa, \tilde{a})$ and $T_1 \leftarrow\!\!\$\ \mathsf{enc}(\kappa, \hat{a})$, and sends $(\mathit{params}, \mathbf{p}_{\mathrm{zt}}, \{u_i\}_{i\in[\kappa+1]}, T_b)$ to the adversary, who returns a bit b'.

The κ-GDDH assumption states that the probability that the adversary outputs $b' = b$ does not exceed $\frac{1}{2}$ by more than an amount that is negligible in λ.

[2] More precisely, since the realization of graded encodings in [GGH13a] are "noisy" encodings over ideal lattices, encodings u_0 and u_1 must be below a given noise bound, so that the distributions of u_0' and u_1' are statistically the same.

B.2 The Bit-Fixing VRF Implemented With Graded Encoding Systems

Construction. Boneh and Waters [BW13] show that their multilinear-group based constrained PRF can be transformed to the setting of graded encoding system. Here, we show that our bit-fixing VRF can also be implemented in graded encoding systems, and note that the transformation of our circuit-constrained VRF works analogously. A scheme that includes a verification procedure (as for VRFs) which is first defined using the abstraction of multilinear groups and then tranformed to graded encodings, is the construction of identity-based aggregate signatures by Hohenberger, Sahai and Waters [HSW13a]. Our graded-encodings-based bit-fixing VRF is defined as follows:

$\underline{\mathsf{Setup}}(1^\lambda, 1^n)$: On input λ and the input length n, Setup runs $\mathsf{InstGen}(1^\lambda, 1^n)$ to generate parameters $(params, \mathbf{p}_{\mathrm{zt}})$ for the graded encoding system. *params* includes a level-1 encoding of $1 \in R$, which we denote as g. Next, run $\mathsf{samp}()$ $2n+1$ times to generate level-0 encodings c and $d_{i,\beta}$ of random elements $\gamma, \delta_{i,\beta} \leftarrow\!\!\$\ R$, for $i \in [n]$, $\beta \in \{0,1\}$. Define $C \leftarrow\!\!\$\ \mathsf{enc}(1, c)$ and $D_{i,\beta} \leftarrow\!\!\$\ \mathsf{enc}(1, d_{i,\beta})$ for $i \in [n]$, $\beta \in \{0,1\}$. The VRF public and secret key are defined as

$$\begin{aligned} pk &:= \big(params, \mathbf{p}_{\mathrm{zt}}, C, \{D_{i,\beta}\}_{i\in[n],\,\beta\in\{0,1\}}\big) \\ sk &:= \big(pk, c, \{d_{i,\beta}\}_{i\in[n],\,\beta\in\{0,1\}}\big) \end{aligned} \tag{3}$$

The domain is $\mathcal{X} = \{0,1\}^n$, the range of the function is $\mathcal{Y} = \{0,1\}^{\ell(\lambda)}$ (with ℓ defined as the output length of the extraction algorithm ext), and proofs are in S_{n-1}.

The function value and proof computation for input $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ are defined as

$$F(sk, x) := \mathsf{ext}\big(\mathbf{p}_{\mathrm{zt}}, \mathsf{enc}(n,\ c \cdot \textstyle\prod_{i\in[n]} d_{i,x_i})\big) \qquad P(sk, x) := \mathsf{reRand}\big(\mathsf{enc}(n-1,\ \textstyle\prod_{i\in[n]} d_{i,x_i})\big)$$

where the product is defined as repeated application of the multiplication algorithm mul to level-0 encodings, as in Equation (2).

$\underline{\mathsf{Verify}}(pk, x, y, \pi)$: To verify a tuple $(x, y, \pi) \in \{0,1\}^n \times \{0,1\}^{\ell(\lambda)} \times S_{n-1}$ w.r.t. a public key pk as in (3), compute $D(x)$, a level-n encoding of $\prod_{i\in[n]} \delta_{i,x_i}$, by multiplying the public-key elements $\{D_{i,x_i}\}_{i\in[n]}$:

$$D(x) \leftarrow\!\!\$\ \mathsf{mul}\Big(n-1, \ldots, \mathsf{mul}\big(2, \mathsf{mul}(1, D_{1,x_1}, 1, D_{2,x_2}), 1, D_{3,x_3}\big), \ldots, 1, D_{n,x_n}\Big) \in S_n^{(\prod_{i\in[n]} \delta_{i,x_i})}$$

First, we check whether the purported proof π is a level-$(n-1)$ encoding of $\prod_{i\in[n]} \delta_{i,x_i}$, that is, whether $\pi \in (S_{n-1})^{(\prod_{i\in[n]} \delta_{i,x_i})}$. We can do this by multiplying π with g (the level-1 encoding of 1 contained in *params*), which for correct proofs yields an element of $S_n^{(\prod_{i\in[n]} \delta_{i,x_i})}$. Subtracting $D(x)$, which is an element of this set, should then yield and encoding of 0, which can be verified using the zero-test for level-n elements.

It then remains to check whether y is the canonical representative of $\gamma \cdot \prod_{i\in[n]} \delta_{i,x_i} \in R$, which can be done by multiplying C and π, which yields an element of $S_n^{(\gamma\cdot\prod_{i\in[n]} \delta_{i,x_i})}$, from which we can extract the canonical representation. Verify thus output 1 if the following equations are satisfied:

$$\mathsf{isZero}\Big(\mathsf{add}\big(n, \mathsf{mul}(1, G, n-1, \pi), \mathsf{neg}(n, D(x))\big)\Big) = 1 \qquad \mathsf{ext}\big(\mathbf{p}_{\mathrm{zt}}, \mathsf{mul}(1, C, n-1, \pi)\big) = y$$

$\underline{\mathsf{Constrain}}(sk, \mathbf{v})$: Since multiplying a proof $P(sk, x)$ with the public-key element C and extracting the canonical representative yields $F(sk, x)$ (cf. the second verification equation), it suffices that a constrained key lets us construct $P(sk, x)$.

$\mathsf{Constrain}$ takes as input sk and a vector $\mathbf{v} \in \{0,1,?\}^n$ describing the constrained input space as $S_{\mathbf{v}} := \{x \in \{0,1\}^n \,|\, \forall i \in [n] : x_i = \mathbf{v}_i \vee \mathbf{v}_i = ?\}$. Let $V := \{i \in [n] \,|\, \mathbf{v}_i \neq ?\}$ be the set of indices fixed by $\mathbf{v}$. Return $sk_{\mathbf{v}} := (pk, k_{\mathbf{v}})$, with $k_{\mathbf{v}}$ defined as follows:

If $V = \{j\}$ then set $k_{\mathbf{v}} \leftarrow\!\!\$\ \mathsf{reRand}(d_{j,\mathbf{v}_j}) \in S_0^{(d_{j,\mathbf{v}_j})}$.

If $|V| > 1$ then repeatedly apply mul to $d_{i,\mathbf{v}_i}$ for all $i \in V$, to compute a level-0 encoding $k'_{\mathbf{v}}$ of $\prod_{i\in V} \delta_{i,\mathbf{v}_i}$ and set $k_{\mathbf{v}} \leftarrow\!\!\$\ \mathsf{reRand}\big(\mathsf{enc}(|V|-1, k_{\mathbf{v}})\big) \in (S_{|V|-1})^{(\prod_{i\in V} \delta_{i,\mathbf{v}_i})}$.

$\underline{\mathsf{Prove}(sk_{\mathbf{v}}, x)}$: Again let $V := \{i \in [n] \,|\, \mathbf{v}_i \neq ?\}$ and let $\overline{V} := \{i \in [n] \,|\, \mathbf{v}_i = ?\}$ be its complement. If $x_i \neq \mathbf{v}_i$ for some $i \in V$ then abort. Otherwise compute a level-$|\overline{V}|$ encoding $D_{\overline{V}}(x)$ of $\prod_{i\in\overline{V}} \delta_{i,x_i}$ by repeatedly applying mul to $\{D_{i,x_i}\}_{i\in\overline{V}}$. Then multiply $D_{\overline{V}}(x)$ with the constrained key $k_{\mathbf{v}}$, which yields a level-$(n-1)$ encoding of $\prod_{i\in[n]} \delta_{i,x_i}$, which we rerandomize to obtain the proof:

$$\pi \leftarrow\!\!\$\ \mathsf{reRand}\big(\mathsf{mul}(|\overline{V}|, D_{\overline{V}}, |V|-1, k_{\mathbf{v}})\big) \ \in\ (S_{n-1})^{(\prod_{i\in[n]} \delta_{i,x_i})} \ .$$

Output $F(sk, x) := \mathsf{ext}\big(\mathbf{p}_{\mathrm{zt}}, \mathsf{mul}(1, C, n-1, \pi)\big)$, the random representative of $\gamma \cdot \prod_{i\in[n]} \delta_{i,x_i} \in R$.

Security. Provability and uniqueness follow from the correctness of the algorithms for the graded encoding system. In contrast to the instantiation based on multilinear groups, there is no unique proof; instead, every level-$(n-1)$ encoding of $\prod_{i\in[n]} \delta_{i,x_i}$ satisfies the verification equation. Constraint-hiding follows form the fact that $\mathsf{Constrain}$ randomizes the proof elements before outputting them.

The proof of pseudorandomness proceeds analogously to that for our instantiation in Section 4. Given an n-GDDH challenge $(params, \mathbf{p}_{\mathrm{zt}}, \{u_i\}_{i\in[n+1]}, T)$, the simulator guesses the challenge query $x^* \leftarrow\!\!\$\ \{0,1\}^n$ and sets the public-key components $D_{i,x_i^*} := u_i$ and $C := u_{n+1}$. For the remaining components, it samples n level-0 elements $z_i \leftarrow\!\!\$\ \mathsf{samp}()$ and sets $D_{i,\overline{x}_i^*} := \mathsf{enc}(1, z_i)$ (where $\overline{x}_i^*$ denotes the bit $1-x_i^*$). The challenge query is answered as $\mathsf{ext}(\mathbf{p}_{\mathrm{zt}}, T)$, while constrained-key and prove queries are simulated using the values z_i as in the proof for our construction based on multilinear groups in Section 4.3.