Hindawi
Security and Communication Networks
Volume 2019, Article ID 4187892, 15 pages
https://doi.org/10.1155/2019/4187892

Research Article

A Novel Construction of Constrained Verifiable Random Functions

Muhua Liu, Ping Zhang, and Qingtao Wu

Control Science and Engineering Postdoctoral Mobile Station, Henan University of Science and Technology, Luoyang 471023, China

Correspondence should be addressed to Muhua Liu; lxk0379@126.com

Received 4 April 2019; Revised 30 August 2019; Accepted 11 October 2019; Published 3 November 2019

Guest Editor: Giovanni Agosta

Constrained verifiable random functions (VRFs) were introduced by Fuchsbauer. In a constrained VRF, one can drive a constrained key sk_S from the master secret key sk, where S is a subset of the domain. Using the constrained key sk_S, one can compute function values at points which are not in the set S. The security of constrained VRFs requires that the VRFs' output should be indistinguishable from a random value in the range. They showed how to construct constrained VRFs for the bit-fixing class and the circuit constrained class based on multilinear maps. Their construction can only achieve selective security where an attacker must declare which point he will attack at the beginning of experiment. In this work, we propose a novel construction for constrained verifiable random function from bilinear maps and prove that it satisfies a new security definition which is stronger than the selective security. We call it semiadaptive security where the attacker is allowed to make the evaluation queries before it outputs the challenge point. It can immediately get that if a scheme satisfied semiadaptive security, and it must satisfy selective security.

1. Introduction

Pseudorandom functions (PRFs) are one of the basic concepts in modern cryptography, which were introduced by Goldreich et al. [1]. A PRF is an efficiently computable function $F : \mathcal{K} \times \mathcal{X} \longrightarrow \mathcal{Y}$. For a randomly chosen key $sk \in \mathcal{K}$, a polynomial probabilistic time (PPT) adversary cannot distinguish the outputs $F(sk, x)$ of the function for any $x \in \mathcal{X}$ from a randomly chosen values from $\mathcal{Y}$.

Boneh and Waters [2] put forward the concept of PRFs and presented a new notion which was called constrained pseudorandom functions. A constrained PRF is the same as the standard PRF except that it is associated with a set $S \subset \mathcal{X}$. It contains a master key $sk \in \mathcal{K}$ which can be used to evaluate all points that belonged to the domain $\mathcal{X}$. Given the master key $sk \in \mathcal{K}$ and a set $S \subset \mathcal{X}$, it can generate a constrained key sk_S which can be used to evaluate $F(sk, x)$ for any $x \notin S$. Pseudorandomness requires that given several constrained keys for sets $S_1, \ldots, S_{q_1} \subset \mathcal{X}$ and several function values at points $x_1, \ldots, x_{q_2} \in \mathcal{X}$ which were chosen adaptively by the adversary, the adversary cannot distinguish a function value $F(sk, x)$ from a random value for all $x \neq x_i$, $\forall i \in \{1, \ldots, q_2\}$, and $x \in \cap_{j=1}^{q_1} S_j$. Constrained PRFs have been used to optimize the ciphertext length of broadcast encryption [2] and construct multiparty key exchange [3].

Verifiable random functions were introduced by Micali et al. [4]. A VRF is similar to a pseudorandom function. It also preserves the pseudorandomness that a PPT adversary cannot distinguish an evaluated value $F(sk, x)$ from a random value even if it is given several values at other points. A VRF has an additional property that the party holding the secret key is allowed to evaluate F on $x \in \mathcal{X}$ associated with a noninteractive proof. With the proof, anyone can verify the correctness of a given evaluation by the public key. In addition, the evaluation of $F(sk, x)$ should remain pseudorandomness and even an adversary can query values and proofs at other points. Lastly, the verification should remain

sound even if the public key was computed maliciously. VRFs have been used to construct zero knowledge proofs [5], and electronic payment schemes [6], and so on.

In SCN 2014, Fuchsbauer [7] extended the notion of VRFs to a new notion, which was called constrained VRFs. In addition to three polynomial time algorithms: Setup, Prove, and Verify, they defined another algorithm Constrain, which was used to drive a constrained key. For constrained VRFs, it generates a pair key (pk, sk) in the Setup algorithm. Given a constrained key sk_S for a set $S \subset \mathcal{X}$, the algorithm Prove computes a value $y = F(sk, x)$ associated with a prove π which can be used to verify the correctness of $y = F(sk, x)$ by the public key pk. A constrained VRF should satisfy the security notions of provability, uniqueness, and pseudorandomness. The pseudorandomness requires that the evaluation of $F(sk, x)$ should be indistinguishable from a random value, even if the adversary is given several constrained keys for subset $S_1, \ldots, S_{q_1} \subset \mathcal{X}$ and several function values associated with proofs at points $x_1, \ldots, x_{q_2}$, where $x \neq x_i$, $\forall i \in [q_2]$, and $x \in \cap_{j=1}^{q_1} S_j$.

A possible application of constrained VRFs is micropayments [8]. Micropayment schemes emphasized the ability to make payments of small amounts. In the micropayment based on probability, a large number of users and merchants jointly select a user to pay the cheque. It can realize the micropayment of a large number of users to be converted into a macropayment of a certain user with a small probability. In this scheme, how do we decide which cheque C should be payable in fair way? Using the VRFs, merchant M publishes pk_M for VRF with range $\mathcal{Y} \in [0, 1]$. Cheque C is payable if $F(sk_M, C) < s$, where s is a known selection rate. However, it has a drawback which needs a public key infrastructure (PKI) for merchants' keys pk_M. By the constrained VRFs, every merchant uses the same key sk. Merchant M gets constrained key sk_M for set (id_M, C), where id_M is the identity of merchant M. Cheque C is payable if $F(sk_M, id_M \| C) < s$. Anybody can check the result by the same public key pk. Therefore, it does not need the PKI for merchants.

Fuchsbauer [7] gave two constructions from the multilinear maps based on constrained PRFs proposed by Boneh and Waters [2]. The first one is bit-fixing VRFs, in which the constrained keys can be derived for any set $S_\upsilon \subset \{0, 1\}^n$, where S_υ is described by a vector $\upsilon \in \{0, 1, \perp\}$ as the set of all strings such that it matches υ at all coordinates that are not $\perp$. The second one is circuit constrained VRFs, in which the constrained keys can be derived for any set that is decidable by a polynomial size circuit.

However, Fuchsbauer's constructions [7] can only achieve selective security—a weaker notion where the adversary must commit to a challenge point x^* at the beginning of the experiment. By the technology of complexity leveraging, any selective security can be converted into adaptive security where the adversary can make its challenge query at any point. The reduction simply guesses beforehand which challenge value the adversary will query. Therefore, it leads to a security loss that is exponential in the input length. In this work, we attempt to ask an ambitious question: is it possible to construct a constrained VRF which satisfies a more stronger security compared with the selective security?

In this work, we propose a novel construction based on the bilinear maps. Inspired by the constrained PRFs of Hohenberber et al. [9], we construct a VRF with constrained keys for any sets of polynomial size and define a new security named semiadaptive security. It allows the adversary to query the evaluation oracle before it outputs a challenge point, while the public key is returned to the adversary associated with the challenge evaluation. This definition is stronger than the selective security, which can be verified easily.

Our scheme is derived from the constructions of constrained PRFs given by Hohenberger et al. [9]. It is defined over a bilinear group, which contains three groups G_1, G_2, and G_T with composite order $N = pq$, equipped with bilinear maps $e : G_1 \times G_2 \longrightarrow G_T$. The constrained VRFs map an input from $\{0, 1\}^{\ell(\lambda)}$ to an element of G_T. The secret key is a tuple $sk = (v, w^\gamma, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}), h)$, where $v \in G_1, w \in G_2, \gamma \longleftarrow \mathbb{Z}_N^2, \{d_{i,0}, d_{i,1}\}_{i=1}^n \longleftarrow \mathbb{Z}_N^2, h : \{0, 1\}^{\ell(\lambda)} \longrightarrow \{0, 1\}^{n(\lambda)}$ is an admissible hash function. VRFs are defined as

$$F(sk, x) := e\left(v^{\prod_{i=1}^n d_{i,h(x)_i}}, w^\gamma\right), \tag{1}$$

associated with a proof

$$P(sk, x) := v^{\prod_{i=1}^n d_{i,h(x)_i}}, \tag{2}$$

where $h(x)_i$ is the ith bit of $h(x)$.

In order to verify the correctness of evaluation, we define a public key as $pk = (w, w^\gamma, i\mathcal{O}(\mathcal{C}))$, where $i\mathcal{O}(\mathcal{C})$ is an obfuscation of a circuit which takes a point x as input and outputs an element $D(x) := e(v^{\prod_{i=1}^n d_{i,h(x)_i}}, w)$ from G_T. The verifier only needs to check $e(P(sk, x), w) = D(x)$ and $e(P(sk, x), w^\gamma) = F(sk, x)$. The constrained key is an obfuscation of a circuit that has the secret key sk and the constrained set S hardwired in it. On input a value $x \notin S$, it outputs $(F(sk, x), P(sk, x))$. While this solution would work only if the obfuscator achieves a black box obfuscation definition [10], there is no reason to believe that an indistinguishability obfuscator would necessarily hide the secret key sk.

We solve this problem by a new technique which was introduced by Hohenberger [9]. We divide the domain into two disjoint sets by the admissible hash function: computable set and challenge set. The proportion of computable set in the domain is about $1 - 1/Q(\lambda)$, and the proportion of challenge set in the domain is about $1/Q(\lambda)$, where $Q(\lambda)$ is the number of queries made by the adversary. In the evaluation queries before the adversary outputs the challenge point, we use the secret key sk to answer the evaluation query x and abort the experiment if x belonged to the challenge set. After the adversary outputs a challenge point x^*, we use a freshly chosen secret key sk' to answer the evaluation queries. Via a hybrid argument, we reduce weak Bilinear Diffie–Hellman Inversion (BDHI) assumption to the pseudorandomness of constrained VRFs.

1.1. Related Works. Lysyanskaya [11] gave a construction of VRFs in bilinear groups, but the size of proofs and keys is linear in input size, which may be undesirable in resource constrained user. Dodis and Yampolskiy [12] gave a simple and efficient construction of VRFs based on bilinear mapping. Their VRFs' proofs and keys have constant size, but it is only suitable for small input spaces. Hohenberger and Waters [13] presented the first VRFs for exponentially large input spaces under a noninteractive assumption. Abdalla et al. [14] showed a relation between VRFs and identity-based key encapsulation mechanisms and proposed a new VRF-suitable identity-based key encapsulation mechanism from the decisional ℓ–weak Bilinear Diffie–Hellman Inversion assumption.

Fuchsbauer et al. [15] studied the adaptive security of the GGM construction for constrained PRFs and gave a new reduction that only loses a quasipolynomial factor $q^{O(\log \lambda)}$, where q is the number of adversary's queries. Hofheinz et al. [16] gave a new constrained PRF construction for circuit that has polynomial reduction to indistinguishability obfuscation in the random oracle model.

Kiayias et al. [17] introduced a novel cryptographic primitive called delegatable pseudorandom function, which enables a proxy to evaluate a pseudorandom function on a strict subset of its domain using a trapdoor derived from the delegatable PRF's secret key. Boyle et al. [18] introduced functional PRFs which can be seen as constrained PRFs. In functional PRFs, in addition to a master secret key, there are other secret keys for a function f, which allows one to evaluate the pseudorandom function on any y for which there exists an x such that $f(x) = y$. Chandran et al. [19] showed constructions of selectively secure constrained VRFs for the class of all polynomial-sized circuits.

Notations. In what follows, we will denote with $\lambda \in \mathbb{N}$ a security parameter. We say $\mathrm{negl}(\lambda)$ is negligible if $|\mathrm{negl}(\lambda)| < 1/\mathrm{poly}(\lambda)$ holds for all polynomials $\mathrm{poly}(\lambda)$ and all sufficiently large λ. Denote PPT as probabilistic polynomial time. Denote $[n]$ as the set $\{1, \ldots, n\}$.

2. Preliminaries

We first give a definition of admissible hash functions which is introduced by Boneh and Boyen [20].

Definition 1 (see [20]). Let ℓ, n, and θ be efficiently computable univariate polynomials. An efficiently computable function $h : \{0,1\}^{\ell(\lambda)} \longrightarrow \{0,1\}^{n(\lambda)}$ and an efficient randomized algorithm AdmSample are θ–admissible if for any $u \in \{0,1,\perp\}^{n(\lambda)}$, define $H_u : \{0,1\}^{\ell(\lambda)} \longrightarrow \{0,1\}$ as follows: $H_u(x) = 0$ if for all $1 \le j \le n(\lambda)$ and $h(x)_j \neq u_j$, else $H_u(x) = 1$. For any efficiently computable polynomial $Q(\lambda)$, $\forall x_1, \ldots, x_{Q(\lambda)}$, $z \in \{0,1\}^{\ell(\lambda)}$, where $z \neq x_i$, $\forall i \in [Q(\lambda)]$, we have that

$$\Pr\left[\forall i \le Q(\lambda), H_u(x_i) = 1 \wedge H_u(z) = 0\right] \ge 1/\theta(Q(\lambda)), \quad (3)$$

where the probability is taken only over $u \longleftarrow \mathrm{AdmSample}(1^\lambda, Q(\lambda))$.

Next, we present the formal definition of indistinguishability obfuscation following the syntax of Garg et al. [21].

Definition 2 (indistinguishability obfuscation (iO)). A uniform PPT machine $i\mathcal{O}$ is called an indistinguishability obfuscator for a circuit class $\{\mathcal{C}_\lambda\}$ if the following holds:

(i) Correctness: for all security parameters $\lambda \in \mathbb{N}$, for all $C \in \mathcal{C}_\lambda$, and for all inputs x, we have

$$\Pr\left[C'(x) = C(x) : C' \longleftarrow i\mathcal{O}(\lambda, C)\right] = 1. \quad (4)$$

(ii) Indistinguishability: for any (not necessarily uniform) PPT distinguisher $\mathrm{Samp}, \mathcal{D}$, there exists a negligible function negl such that the following holds: if $\Pr[\forall x, C_0(x) = C_1(x); (C_0, C_1, \sigma) \longleftarrow \mathrm{Samp}(1^\lambda)] > 1 - \mathrm{negl}(\lambda)$, then

$$\begin{aligned}&\left|\Pr\left[\mathcal{D}(\sigma, i\mathcal{O}(\lambda, C_0)) = 1 : (C_0, C_1, \sigma) \longleftarrow Samp\left(1^\lambda\right)\right]\right.\\ &\left. - \Pr\left[\mathcal{D}(\sigma, i\mathcal{O}(\lambda, C_1)) = 1 : (C_0, C_1, \sigma) \longleftarrow Samp\left(1^\lambda\right)\right]\right|\\ &\le \mathrm{negl}(\lambda).\end{aligned} \quad (5)$$

2.1. Assumptions. Let $\mathcal{G}$ be a PPT group algorithm that takes a security parameter 1^λ as input and outputs as tuple $(N, G_p, G_q, G_1, G_2, G_T, e)$, in which p and q are independent uniform random λ–bit primes. G_1, G_2, and G_T are groups of order $N = pq$, $e : G_1 \times G_2 \longrightarrow G_T$ is a bilinear map, and G_p and G_q are the subgroups of G_1 with the order p and q, respectively.

The subgroup decision assumption [22] in the bilinear group is said that the uniform distribution on G_1 is computationally indistinguishable from the uniform distribution on a subgroup of G_p or G_q.

Assumption 1 (subgroup hiding for composite order bilinear groups). Let $(N, G_p, G_q, G_1, G_2, G_T, e) \longleftarrow \mathcal{G}(1^\lambda)$ and $b \longleftarrow \{0,1\}$. Let $T \longleftarrow G_1$ if $b = 0$, else $T \longleftarrow G_p$. The advantage of algorithm $\mathcal{A}$ in solving the subgroup decision problem is defined as

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{SGH}} = \left|\Pr\left[b \longleftarrow \mathcal{A}\left(N, G_p, G_q, G_1, G_2, G_T, e, T\right)\right] - 1/2\right|. \quad (6)$$

We say that the subgroup decision problem is hard if for all PPT $\mathcal{A}$, $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{SGH}}$ is negligible in λ.

Assumption 2 (weak Bilinear Diffie–Hellman Inversion). Let $(N, G_p, G_q, G_1, G_2, G_T, e) \longleftarrow \mathcal{G}(1^\lambda)$, $g_1 \longleftarrow G_1, a \longleftarrow \mathbb{Z}_N^*$, and $g_2 \longleftarrow G_2, \gamma \longleftarrow \mathbb{Z}_N^*$. Let $D = (N, G_p, G_q, G_1, G_2, G_T, e, g_1, g_1^a, \ldots, g_1^{a^{n-1}}, g_2, g_2^\gamma)$. Let $T = e(g_1^{a^n}, g_2^\gamma)$ if $b = 0$, else

$T \longleftarrow G_T$. The advantage of algorithm $\mathcal{A}$ in solving the problem is defined as

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{BDHI}} = |\Pr[b \longleftarrow \mathcal{A}(D, T)] - 1/2|. \quad (7)$$

We say that the weak bilinear Diffie–Hellman inversion problem is hard if for all PPT $\mathcal{A}$, $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{BDHI}}$ is negligible in λ.

Chase et al. [22] showed that many q–type assumptions can be implied by subgroup hiding in bilinear groups of composite order.

3. Definition

We recall the definition of constrained VRFs which was given by Fuchsbauer [7].

Let $F : \mathcal{K} \times \mathcal{X} \longrightarrow \mathcal{Y}$ be an efficiently computable function, where $\mathcal{K}$ is the key space, $\mathcal{X}$ is the input domain, and $\mathcal{Y}$ is the range. F is said to be constrained VRFs with regard to a set $S \subset \mathcal{X}$ if there exists a constrained key space $\mathcal{K}'$, a proof space $\mathcal{P}$, and four algorithms (Setup, Constrain, Prove, and Verify) :

(i) $\mathrm{Setup}(1^\lambda) \longrightarrow (pk, sk)$: it is a PPT algorithm that takes the security parameter λ as input and outputs a pair of keys (pk, sk), a description of the key space $\mathcal{K}$, and a constrained key space $\mathcal{K}'$

(ii) $\mathrm{Constrain}(sk, S) \longrightarrow sk_S$: this algorithm takes the secret key sk and a set $S \subset \mathcal{X}$ as input and outputs a constrained key $sk_S \in \mathcal{K}'$

(iii) $\mathrm{Prove}(sk_S, x) \longrightarrow (y, \pi)$ or $(\perp, \perp)$: this algorithm takes the constrained key sk_S and a value x as input and outputs a pair $(y, \pi) \in \mathcal{Y} \times \mathcal{P}$ of a function value and a proof if $x \notin S$, else outputs $(\perp, \perp)$

(iv) $\mathrm{Verify}(pk, x, y, \pi) \longrightarrow \{0, 1\}$: this algorithm takes the public key pk, an input x, a function value y, and a proof π as input and outputs a value in $\{0, 1\}$, where "1" indicates that $y = F(sk, x)$

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

(i) If $x \notin S$, then $y = F(sk, x)$ and $\mathrm{Verify}(pk, x, y, \pi) = 1$

(ii) If $x \in S$, then $(y, \pi) = (\perp, \perp)$

3.2. Uniqueness. For all $\lambda \in \mathbb{N}$, $(pk, sk) \longleftarrow \mathrm{Setup}(1^\lambda)$, $x \in \mathcal{X}$, $y_0, y_1 \in \mathcal{Y}$, and $\pi_0, \pi_1 \in \mathcal{P}$, one of the following conditions holds:

(i) $y_0 = y_1$,

(ii) $\mathrm{Verify}(pk, x, y_0, \pi_0) = 1$, or

(iii) $\mathrm{Verify}(pk, x, y_1, \pi_1) = 1$,

which implies that for every x there is only one value y such that $F(sk, x) = y$.

3.3. Pseudorandomness. We consider the following experiment $\mathrm{Exp}_{\mathcal{A}}^{\mathrm{VRF}}(1^\lambda, b)$ for $\lambda \in \mathbb{N}$:

(i) The challenger first chooses $b \longleftarrow \{0, 1\}$ and then generates (pk, sk) by running the algorithm $\mathrm{Setup}(1^\lambda)$ and returns pk to the adversary $\mathcal{A}$

(ii) The challenger initializes two sets V and E and sets $V := \emptyset, E := \emptyset$, where V will contain the points that the adversary $\mathcal{A}$ cannot evaluate and E contains the points at which the adversary queries the evaluation oracle

(iii) The adversary $\mathcal{A}$ is given the following oracle:

(1) Constrain: on input a set $S \subset \mathcal{X}$, if $V \cap S \neq \emptyset$, return $sk_S \longleftarrow \mathrm{Constrain}(sk, S)$ and set $V := V \cap S$; else return $\perp$

(2) Evaluation: given $x \in \mathcal{X}$, return $(F(sk, x)$ and $P(sk, x))$ and set $E := E \cup \{x\}$

(3) Challenge: on input $x^* \in \mathcal{X}$, if $x^* \in E$ or $x^* \notin V$, then it returns $\perp$. Else, it returns $F(sk, x^*)$ if $b = 0$, or it returns a random value from $\mathcal{Y}$ if $b = 1$

(iv) $\mathcal{A}$ outputs a bit b'; if $b = b'$, the experiment outputs 1

A constrained VRF is pseudorandomness if for all PPT adversary $\mathcal{A}$, it holds that

$$\left|\Pr\left[\mathrm{Exp}_{\mathcal{A}}^{\mathrm{VRF}}\left(1^\lambda, b\right) = 1\right] - 1/2\right| \leq \mathrm{negl}(\lambda). \quad (8)$$

3.4. Semiadaptive Security. We give a weak definition for pseudorandomness which is called semiadaptive security. It allows the adversary to query the evaluation before it outputs a challenge point, while the public key is returned to the adversary after the adversary commits a challenge point. In the selective security, the adversary must commit a challenge input at the beginning of the experiment. Therefore, we can find that if a scheme satisfies the semiadaptive security, it must satisfy selective security. Conversely, it may be not true.

3.5. Puncturable Verifiable Random Functions. Puncturable VRFs are a special class of constrained VRFs, in which the constrained set contains only one value, i.e., $S = \{x^*\}$. The properties of provability, uniqueness, and pseudorandomness are similar to the constrained VRFs. To avoid repetition, we omit the formal definitions.

4. Construction

In this section, we give our construction for puncturable VRFs. A puncturable VRF $F : \mathcal{K} \times \mathcal{X} \longrightarrow \mathcal{Y}$ consists of four algorithms (Setup, Puncture, Prove, and Verify). The input domain is $\mathcal{X} \longleftarrow \{0, 1\}^\ell$, where $\ell = \ell(\lambda)$. The key space $\mathcal{K}$ and range space $\mathcal{Y}$ are defined as a part of the setup algorithm.

(i) $\mathrm{Setup}(1^\lambda) \longrightarrow (pk, sk)$: On input the security parameter 1^λ, run $(N, G_p, G_q, G_1, G_2, G_T, e) \longleftarrow \mathcal{G}(1^\lambda)$ such that $e : G_1 \times G_2 \longrightarrow G_T$ and G_p and G_q

are subgroups of G_1. Let n, θ be polynomials such that there exists a θ–admissible hash function $h : \{0,1\}^{\ell(\lambda)} \longrightarrow \{0,1\}^{n(\lambda)}$.

The key space is $\mathcal{K} = G_1 \times G_2 \times \mathbb{Z}_N \times (\mathbb{Z}_N^2)^n$, the range is $\mathcal{Y} = G_T$, and the proof space is $\mathcal{P} = G_1$. The setup algorithm chooses $v \in G_1, w \in G_2, \gamma \in \mathbb{Z}_N$, and $(d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}) \in \mathbb{Z}_N^2$ uniformly at random and sets $sk = (v, w^\gamma, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$. The public key contains an obfuscation of a circuit $\mathscr{C}_1$, where $\mathscr{C}_1$ is described in Figure 1. Note that $\mathscr{C}_1$ has $v, w, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}), h$ hardwired in it. Set $pk = (w, w^\gamma, i\mathcal{O}(\mathscr{C}_1))$, where $\mathscr{C}_1$ is padded to be of appropriate size. The puncturable VRF F is defined as follows. Let $h(x) = b_1, \ldots, b_n$, where $b_i \in \{0,1\}$. Then,

$$F(sk, x) = e\left(v^{\prod_{j=1}^{n} d_{j,b_j}}, w^\gamma\right), P(sk, x) = v^{\prod_{j=1}^{n} d_{j,b_j}}. \tag{9}$$

(ii) Puncture$(sk, x') \longrightarrow sk_{x'}$: This algorithm computes an obfuscation of a circuit $\mathscr{C}_2$ which is defined in Figure 2. Note that $\mathscr{C}_2$ has the secret key sk and the puncturable value x' hardwired in it. Set $sk_{x'} \longleftarrow i\mathcal{O}(\mathscr{C}_2)$ where $\mathscr{C}_2$ is padded to be of appropriate size.

(iii) Prove$(sk_{x'}, x) \longrightarrow (y, \pi)$ or $(\perp, \perp)$: The punctured key $sk_{x'}$ is a program that takes an ℓ–bit input x. We define

$$\text{Prove}(sk_{x'}, x) = sk_{x'}(x). \tag{10}$$

(iv) Verify$(pk, x, y, \pi) \longrightarrow \{0,1\}$: To verify $(x, y, \pi) \in \{0,1\}^{\ell(\lambda)} \times G_T \times G_1$ with regard to $pk = (w, w^\gamma, i\mathcal{O}(\mathscr{C}_1))$, compute $D(x) \coloneqq \mathscr{C}_1(x) = e(v^{\prod_{j=1}^{n} d_{j,b_j}}, w)$ and output 1 if the following equations are satisfied:

$$e(\pi, w) = D(x), \quad e(\pi, w^\gamma) = y. \tag{11}$$

4.1. Properties

4.1.1. Provability. From the definition of F and P, we observe that for $(pk, sk) \longleftarrow \text{Setup}(1^\lambda)$, $x \in \mathcal{X}, sk_{x'} \longleftarrow \text{Puncture}(sk, x')$, $(y, \pi) = \text{Prove}(sk_{x'}, x)$, if $x \neq x'$:

$$\begin{aligned} e(\pi, w) &= e\left(v^{\prod_{j=1}^{n} d_{j,b_j}}, w\right) = D(x), \\ e(\pi, w^\gamma) &= e\left(v^{\prod_{j=1}^{n} d_{j,b_j}}, w^\gamma\right) = y = F(sk, x). \end{aligned} \tag{12}$$

Therefore, we have Verify$(pk, x, y, \pi) = 1$. When $x = x'$, we can get that Prove$(sk_{x'}, x') = (\perp, \perp)$. This completes the proof of provability.

4.1.2. Uniqueness. Consider a public key $pk = (w, w^\gamma, i\mathcal{O}(\mathscr{C}_1))$, where $w \in G_2, \gamma \in \mathbb{Z}_N$, and $\mathscr{C}_1$ is described in Figure 1. Given a value $x \in \{0,1\}^{\ell(\lambda)}$ and two pair values (y_0, π_0) and $(y_1, \pi_1) \in G_T \times G_1$ that satisfy Verify$(pk, x, y_b, \pi_b) = 1$ for $b \in \{0,1\}$. We show that $y_0 = y_1$.

Input: a value x
Constants: $v, w, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}), h$

(1) compute $h(x) = b_1, \ldots, b_n$;
(2) output $D(x) = e(v^{\prod_{j=1}^{n} d_{j,b_j}}, w)$.

Figure 1: Circuit $\mathscr{C}_1$.

Input: a value x
Constants: the secret key sk, the puncturable tvalue $x' \in \{0,1\}^{\ell(\lambda)}$

(1) compute $h(x) = b_1, \ldots, b_n$;
(2) if $x = x'$, output $(\perp, \perp)$;
(3) else,output $y = e(v^{\prod_{j=1}^{n} d_{j,b_j}}, w^\gamma)$, $\pi = v^{\prod_{j=1}^{n} d_{j,b_j}}$.

Figure 2: Circuit $\mathscr{C}_2$.

From the verification equations, we observe that $e(\pi_b, w) = D(x)$ and $e(\pi_b, w^\gamma) = y_b$. Because $D(x) = e(v^{\prod_{j=1}^{n} d_{j,b_j}}, w)$, then $\pi_0 = \pi_1$. Therefore, we can get that $y_0 = e(\pi_0, w^\gamma) = e(\pi_1, w^\gamma) = y_1$.

4.2. Proof of Pseudorandomness. In this section, we prove that our construction is secure puncturable VRFs as defined in Section 3.

Theorem 1. *Assuming $i\mathcal{O}$ is a secure indistinguishability obfuscator and the subgroup hiding assumption for composite order bilinear groups holds, then our construction described as above satisfies the semiadaptive security as defined in Section 3.*

Proof. To prove the above theorem, we first define a sequence of games where the first one is the original pseudorandomness security game and show that each adjacent games is computationally indistinguishable for any PPT adversary $\mathscr{A}$. Without loss of generality, we assume that the adversary $\mathscr{A}$ makes $Q = Q(\lambda)$ evaluation queries before outputting the challenge point, where $Q(\lambda)$ is a polynomial. We present a full description of each game and underline the changes from the present one to the previous one. Each such game is completely characterized by its key generation algorithm and its challenge answer. The differences between these games are summarized in Table 1. □

4.2.1. Game 1. The first game is the original security for our construction. Here, the challenger first chooses a puncturable VRF key. Then, $\mathscr{A}$ makes evaluation queries and finally outputs a challenge point. The challenger responds with either a PRF evaluation or a random value.

(1) The challenger runs $(N, G_p, G_q, G_1, G_2, G_T, e) \longleftarrow \mathcal{G}(1^\lambda)$, chooses $v \in G_1, w \in G_2, \gamma \in \mathbb{Z}_N, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}) \in \mathbb{Z}_N^2$, and sets $sk = (v, w, \gamma, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$ and $pk = (w, w^\gamma, i\mathcal{O}(\mathscr{C}_1))$. Then, the challenger flips a coin $\alpha \longleftarrow \{0,1\}$.

Table 1: The differences between each adjacent games.

Game	Key generation	Challenge answer
Game 1	$sk = (v, w, \gamma, \{(d_{i,0}, d_{i,1})\}_{i=1}^{n})$	$y_0 = e(v^{\prod_{j=1}^{n} d_{j,b_j^*}}, w^{\gamma})$
Game 2	$sk = (v, w, \gamma, \{(d_{i,0}, d_{i,1})\}_{i=1}^{n})$	If $H_u(x^*) = 1$, abort, else $y_0 = e(v^{\prod_{j=1}^{n} d_{j,b_j^*}}, w^{\gamma})$
Game 3	If $H_u(x) = 1$, $sk = (v, w, \gamma, \{(d_{i,0}, d_{i,1})\}_{i=1}^{n})$. Else, $sk' = (v_1, w, \gamma, \{(e_{i,0}, e_{i,1})\}_{i=1}^{n})$	If $H_u(x^*) = 1$, abort, else $y_0 = e(v_1^{\prod_{j=1}^{n} e_{j,b_j^*}}, w^{\gamma})$
Game 4	If $H_u(x) = 1$, $sk = (v, w, \gamma, \{(d_{i,0}, d_{i,1})\}_{i=1}^{n})$. Else, $sk' = (v_1, w, \gamma, \{(e_{i,0}, e_{i,1})\}_{i=1}^{n})$ where $e_{i,b} = e'_{i,0} \cdot a$, if $H_u(x^*)_i = b$	If $H_u(x^*) = 1$, abort, else $y_0 = e(v_1^{\prod_{j=1}^{n} e_{j,b_j^*}}, w^{\gamma})$
Game 5	If $H_u(x) = 1$, $sk = (v, w, \gamma, \{(d_{i,0}, d_{i,1})\}_{i=1}^{n})$. Else, $sk' = (V, w, \gamma, \{(e'_{i,0}, e'_{i,1})\}_{i=1}^{n})$ where $V = (v_1, v_1^{a}, \ldots, v_1^{a^{n-1}})$	If $H_u(x^*) = 1$, abort, else $y_0 = e(v_1^{a^n \prod_{j=1}^{n} e'_{j,b_j^*}}, w^{\gamma})$
Game 6	If $H_u(x) = 1$, $sk = (v, w, \gamma, \{(d_{i,0}, d_{i,1})\}_{i=1}^{n})$. Else, $sk' = (V, w, \gamma, \{(e'_{i,0}, e'_{i,1})\}_{i=1}^{n})$ where $V = (v_1, v_1^{a}, \ldots, v_1^{a^{n-1}})$	If $H_u(x^*) = 1$, abort, else $y_0 = T^{\prod_{j=1}^{n} e'_{j,b_j^*}}$

(2) The adversary $\mathcal{A}$ makes a evaluation query $x_i \in \{0,1\}^{\ell(\lambda)}$. Then, the challenger computes $h(x_i) = b_1^i, \ldots, b_n^i$ and outputs $(F(sk, x_i), P(sk, x_i)) = (e(v^{\prod_{j=1}^{n} d_{j,b_j^i}}, w^{\gamma}), v^{\prod_{j=1}^{n} d_{j,b_j^i}})$.

(3) The adversary $\mathcal{A}$ sends a challenge point x^* such that $x^* \neq x_i$ for all $i \in [Q(\lambda)]$.

(4) The challenger computes $sk_{x^*} \longleftarrow i\mathcal{O}(\mathcal{C}_2)$ and $h(x^*) = b_1^*, \ldots, b_n^*$, sets $y_0 = e(v^{\prod_{j=1}^{n} d_{j,b_j^*}}, w^{\gamma})$ and $y_1 \longleftarrow G_T$, and returns (pk, sk_{x^*}, y_α) to the adversary $\mathcal{A}$.

(5) The adversary $\mathcal{A}$ outputs a bit $\alpha\prime$ and wins if $\alpha' = \alpha$.

4.2.2. Game 2. This game is the same as the Game 1 except that a partitioning game is simulated. If the undesirable partition is queried, we abort the game. The partition game is defined as follows: the challenger samples a string $u \in \{0, 1, \perp\}^n$ by the algorithm AdmSample of admissible hash function and aborts if either there exists a evaluation query x such that $H_u(x) = 0$ or the challenge query x^* such that $H_u(x^*) = 1$.

(1) The challenger runs $(N, G_p, G_q, G_1, G_2, G_T, e) \longleftarrow \mathcal{G}(1^{\lambda})$, chooses $v \in G_1, w \in G_2, \gamma \in \mathbb{Z}_N$, $(d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}) \in \mathbb{Z}_N^2$, and sets $sk = (v, w, \gamma, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$ and $pk = (w, w^{\gamma}, i\mathcal{O}(\mathcal{C}_1))$. Then, the challenger flips a coin $\alpha \longleftarrow \{0,1\}$ and runs $u \longleftarrow \mathrm{AdmSample}(1^{\lambda}, Q)$.

(2) The adversary $\mathcal{A}$ makes a evaluation query $x_i \in \{0,1\}^{\ell(\lambda)}$. The challenger checks if $H_u(x_i) = 1$ (recall that $H_u(x) = 0$ if $h(x)_j \neq u_j$ for all $j \in [n]$). If not, the game aborts. Else, the challenger computes $h(x_i) = b_1^i, \ldots, b_n^i$ and outputs $(F(sk, x_i), P(sk, x_i)) = (e(v^{\prod_{j=1}^{n} d_{j,b_j^i}}, w^{\gamma}), v^{\prod_{j=1}^{n} d_{j,b_j^i}})$.

(3) The adversary $\mathcal{A}$ sends a challenge point x^* such that $x^* \neq x_i$ for all $i \in [Q(\lambda)]$.

(4) The challenger checks if $H_u(x^*) = 0$. If not, the game aborts. Else, the challenger computes $sk_{x^*} \longleftarrow i\mathcal{O}(\mathcal{C}_2)$ and $h(x^*) = b_1^*, \ldots, b_n^*$, sets $y_0 = e(v^{\prod_{j=1}^{n} d_{j,b_j^*}}, w^{\gamma})$ and $y_1 \longleftarrow G_T$, and returns (pk, sk_{x^*}, y_α) to the adversary $\mathcal{A}$.

(5) The adversary $\mathcal{A}$ outputs a bit α' and wins if $\alpha' = \alpha$.

Lemma 1. *For any PPT adversary $\mathcal{A}$, if $\mathcal{A}$ wins with advantage ϵ in Game 1, then it wins with advantage $\epsilon/\theta(Q(\lambda))$ in Game 2.*

Proof. The difference between Game 1 and Game 2 is that we add an abort condition in Game 2. From the θ−admissible of hash function h, we can get that for all $x_1, \ldots, x_Q, x^*$, $\Pr_{u \longleftarrow \mathrm{AdmSample}(1^{\lambda}, Q(\lambda))}[\forall i, H_u(x_i) = 1 \wedge H_u(x^*) = 0] \geq 1/\theta(Q(\lambda))$. The two experiments are equal if Game 2 does not abort. Therefore, if $\mathcal{A}$ wins with advantage ϵ in Game 1, then it wins with advantage at least $\epsilon/\theta(Q(\lambda))$ in Game 2. □

4.2.3. Game 3. This game is the same as the previous one except that the public key and the punctured key are obfuscation of two other circuits defined in Figures 3 and 4, respectively. On inputs x such that $H_u(x) = 1$, the public key and the punctured key use the same secret key sk as before. However, if $H_u(x) = 0$, the public key and the punctured key use a different secret key $sk\prime$ which is randomly chosen from the key space. The detailed description is given as follows:

(1) The challenger runs $(N, G_p, G_q, G_1, G_2, G_T, e) \longleftarrow \mathcal{G}(1^{\lambda})$, chooses $v \in G_1, w \in G_2, \gamma \in \mathbb{Z}_N$, and $(d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}) \in \mathbb{Z}_N^2$, and sets $sk = (v, w, \gamma, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$ and $pk = (w, w^{\gamma}, i\mathcal{O}(\mathcal{C}_1))$. Then, the challenger flips a coin $\alpha \longleftarrow \{0,1\}$ and runs $u \longleftarrow \mathrm{AdmSample}(1^{\lambda}, Q)$.

(2) The adversary $\mathcal{A}$ makes a evaluation query $x_i \in \{0,1\}^{\ell(\lambda)}$. The challenger checks if $H_u(x_i) = 1$ (recall that $H_u(x) = 0$ if $h(x)_j \neq u_j$ for all $j \in [n]$). If not, the game aborts. Else, the challenger computes

Input: a value x
Constants: v, w^{γ}, $(d_{1,0}, d_{1,1})$, ...,$(d_{n,0}, d_{n,1})$, h, the secret key $sk' = (v_1, w, \gamma, (e_{1,0}, e_{1,1}), \ldots, (e_{n,0}, e_{n,1}))$, and the random string $u \in \{0, 1, \perp\}^n$

(a) compute $h(x) = b_1, \ldots, b_n$;
(b) if $H_u(x) = 0$, then output $D(x) = e(v_1^{\prod_{j=1}^{n} d_{j,b_j}}, w)$;
(c) else, output $D(x) = e(v^{\prod_{j=1}^{n} d_{j,b_j}}, w)$

Figure 3: Circuit $\mathscr{C}_3$.

Input: a value x
Constants: the secret key sk, the puncturable value $x^* \in \{0, 1\}^{\ell(\lambda)}$, the secret key $sk' = (v_1, w, \gamma, (e_{1,0}, e_{1,1}), \ldots, (e_{n,0}, e_{n,1}))$, and the random string $u \in \{0, 1, \perp\}^n$

(a) compute $h(x) = b_1, \ldots, b_n$;
(b) if $x = x^*$, output $(\perp, \perp)$;
(c) else if $H_u(x) = 0$, output $y = e(v^{\prod_{j=1}^{n} e_{j,b_j}}, w^{\gamma})$, $\pi = v_1^{\prod_{j=1}^{n} e_{j,b_j}}$;
(d) if $H_u(x) = 1$, output $y = e(v^{\prod_{j=1}^{n} d_{j,b_j}}, w^{\gamma})$, $\pi = v^{\prod_{j=1}^{n} d_{j,b_j}}$.

Figure 4: Circuit $\mathscr{C}_4$.

$h(x_i) = b_1^i, \ldots, b_n^i$ and outputs $(F(sk, x_i), P(sk, x_i)) = (e(v^{\prod_{j=1}^{n} d_{j,b_j^i}}, w^{\gamma}), v^{\prod_{j=1}^{n} d_{j,b_j^i}})$.

(3) The adversary $\mathscr{A}$ sends a challenge point x^* such that $x^* \neq x_i$ for all $i \in [Q(\lambda)]$.

(4) The challenger checks if $H_u(x^*) = 0$. If not, the game aborts. Else, the challenger chooses $v_1 \in G_1$, $(e_{1,0}, e_{1,1}), \ldots, (e_{n,0}, e_{n,1}) \in \mathbb{Z}_N^2$, sets $sk' = (v_1, w, \gamma, (e_{1,0}, e_{1,1}), \ldots, (e_{n,0}, e_{n,1}))$, computes $pk' = (w, w^{\gamma}, i\mathcal{O}(\mathscr{C}_3))$, $sk_{x^*} \longleftarrow i\mathcal{O}(\mathscr{C}_4)$ and $h(x^*) = b_1^*, \ldots, b_n^*$, and sets $y_0 = e(v_1^{\prod_{j=1}^{n} e_{j,b_j^*}}, w^{\gamma})$ and $y_1 \longleftarrow G_T$. Then, it returns $(pk', sk_{x^*}, y_{\alpha})$ to the adversary $\mathscr{A}$.

(5) The adversary $\mathscr{A}$ outputs a bit α' and wins if $\alpha' = \alpha$.

Lemma 2. *Assuming $i\mathcal{O}$ is a secure indistinguishability obfuscator and the assumption 1 holds, Game 2 and Game 3 are computationally indistinguishable.*

This proof is given in Section 4.3.

4.2.4. Game 4. This game is the same as the previous one except that the generation way of secret key sk' is different. We make some elements of secret key sk' to contain a factor a, for use on inputs x where $H_u(x) = 0$. The detailed description is given as follows:

(1) The challenger runs $(N, G_p, G_q, G_1, G_2, G_T, e) \longleftarrow \mathcal{G}(1^{\lambda})$, chooses $v \in G_1, w \in G_2, \gamma \in \mathbb{Z}_N$, $(d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}) \in \mathbb{Z}_N^2$, and sets $sk = (v, w, \gamma, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$ and $pk = (w, w^{\gamma}, i\mathcal{O}(\mathscr{C}_1))$. Then, the challenger flips a coin $\alpha \longleftarrow \{0, 1\}$ and runs $u \longleftarrow \mathrm{AdmSample}(1^{\lambda}, Q)$.

(2) The adversary $\mathscr{A}$ makes a evaluation query $x_i \in \{0, 1\}^{\ell(\lambda)}$. The challenger checks if $H_u(x_i) = 1$ (recall that $H_u(x) = 0$ if $h(x)_j \neq u_j$ for all $j \in [n]$). If not, the game aborts. Else, the challenger computes $h(x_i) = b_1^i, \ldots, b_n^i$ and outputs $(F(sk, x_i), P(sk, x_i)) = (e(v^{\prod_{j=1}^{n} d_{j,b_j^i}}, w^{\gamma}), v^{\prod_{j=1}^{n} d_{j,b_j^i}})$.

(3) The adversary $\mathscr{A}$ sends a challenge point x^* such that $x^* \neq x_i$ for all $i \in [Q(\lambda)]$.

(4) The challenger checks if $H_u(x^*) = 0$. If not, the game aborts. Else, the challenger chooses $v_1 \in G_1$, $a \in \mathbb{Z}_N$, $(e'_{1,0}, e'_{1,1}), \ldots, (e'_{n,0}, e'_{n,1}) \in \mathbb{Z}_N^2$. Set $e_{i,b} = e'_{i,b} \cdot a$, if $h(x^*)_i = b$, else $e_{i,b} = e'_{i,b}$. Let $sk' = (v_1, w, \gamma, (e_{1,0}, e_{1,1}), \ldots, (e_{n,0}, e_{n,1}))$, compute $pk' = (w, w^{\gamma}, i\mathcal{O}(\mathscr{C}_3))$, $sk_{x^*} \longleftarrow i\mathcal{O}(\mathscr{C}_4)$, and $h(x^*) = b_1^*, \ldots, b_n^*$ and set $y_0 = e(v_1^{\prod_{j=1}^{n} e_{j,b_j^*}}, w^{\gamma})$ and $y_1 \longleftarrow G_T$. Then, it returns $(pk', sk_{x^*}, y_{\alpha})$ to the adversary $\mathscr{A}$.

(5) The adversary $\mathscr{A}$ outputs a bit α' and wins if $\alpha' = \alpha$.

Lemma 3. *The outputs of Game 3 and Game 4 are statistically indistinguishable.*

Proof. Recall that the difference between Game 3 and Game 4 is the manner in which $\{e_{i,b}\}_{i \in [n], b \in \{0,1\}}$ are chosen. In Game 3, $\{e_{i,b}\}_{i \in [n], b \in \{0,1\}}$ are chosen randomly from $\mathbb{Z}_N$, while in Game 4, the challenger first chooses $e'_{i,b} \longleftarrow \mathbb{Z}_N$ and $a \longleftarrow \mathbb{Z}_N$ and sets $e_{i,b} = e'_{i,b} \cdot a$, if $h(x^*)_i = b$, else $e_{i,b} = e'_{i,b}$. Since $a \in \mathbb{Z}_N$ which is invertible with overwhelming probability, $e_{i,b} = e'_{i,b} \cdot a$ is a uniform element in $\mathbb{Z}_N$. Hence, the two experiments are statistically indistinguishable. □

4.2.5. Game 5. This game is the same as the previous one except that the hardwire of circuits $\mathscr{C}_3$ and $\mathscr{C}_4$ is changed. The two circuits contain some constants $\{v_1^{a^i}\}_{i=1}^{n-1}$. When $H_u(x) = 0$, the related function values are computed using

the constants $\{v_1^{a^i}\}_{i=1}^n$. The detailed description is given as follows:

(1) The challenger runs $(N, G_p, G_q, G_1, G_2, G_T, e) \longleftarrow \mathcal{G}(1^\lambda)$, chooses $v \in G_1, w \in G_2, \gamma \in \mathbb{Z}_N$, and $(d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}) \in \mathbb{Z}_N^2$, and sets $sk = (v, w, \gamma, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$ and $pk = (w, w^\gamma, i\mathcal{O}(\mathcal{C}_1))$. Then, the challenger flips a coin $\alpha \longleftarrow \{0,1\}$ and runs $u \longleftarrow \mathrm{AdmSample}(1^\lambda, Q)$.

(2) The adversary $\mathcal{A}$ makes an evaluation query $x_i \in \{0,1\}^{\ell(\lambda)}$. The challenger checks if $H_u(x_i) = 1$ (recall that $H_u(x) = 0$ if $h(x)_j \neq u_j$ for all $j \in [n]$). If not, the game aborts. Else, the challenger computes $h(x_i) = b_1^i, \ldots, b_n^i$ and outputs $(F(sk, x_i), P(sk, x_i)) = (e(v^{\prod_{j=1}^n d_{j,b_j^i}}, w^\gamma), v^{\prod_{j=1}^n d_{j,b_j^i}})$.

(3) The adversary $\mathcal{A}$ sends a challenge point x^* such that $x^* \neq x_i$ for all $i \in [Q(\lambda)]$.

(4) The challenger checks if $H_u(x^*) = 0$. If not, the game aborts. Else, the challenger chooses $v_1 \in G_1, a \in \mathbb{Z}_N, (e'_{1,0}, e'_{1,1}), \ldots, (e'_{1,0}, e'_{n,1}) \in \mathbb{Z}_N^2$. Set $\overrightarrow{V} = (v_1, v_1^a, \ldots, v_1^{a^{n-1}})$. Let $sk' = ((v_1, w, \gamma, (e_{1,0}, e_{1,1}), \ldots, (e_{n,0}, e_{n,1}))$, compute $pk' = (w, w^\gamma, i\mathcal{O}(\mathcal{C}_5))$, $sk_{x^*} \longleftarrow i\mathcal{O}(\mathcal{C}_6)$, $h(x^*) = b_1^*, \ldots, b_n^*$, and $D_{x^*} = e(v_1^{a^n \prod_{j=1}^n e'_{j,b_j^*}}, w)$. Set $y_0 = e(v_1^{a^n \prod_{j=1}^n e'_{j,b_j^*}}, w)$ and $y_1 \longleftarrow G_T$, where the descriptions of $\mathcal{C}_5$ and $\mathcal{C}_6$ are given in Figures 5 and 6, respectively. Then, it returns $(pk', sk_{x^*}, y_\alpha)$ to the adversary $\mathcal{A}$.

(5) The adversary $\mathcal{A}$ outputs a bit α' and wins if $\alpha' = \alpha$.

Lemma 4. *Assuming $i\mathcal{O}$ is a secure indistinguishability obfuscator, Game 4 and Game 5 are computationally indistinguishable.*

Proof. We will introduce an intermediate experiment 4_A and prove that Game 4 and 4_A are computationally indistinguishable and Game 4_A and Game 5 are computationally indistinguishable.

The experiment 4_A is the same as Game 5 except that sk_{x^*} is generated by obfuscating the circuit $\mathcal{C}_4$ in Step 4. Assume that there exists a PPT adversary $\mathcal{A}$ that distinguishes the outputs of Game 4 and Game 4_A, we construct a PPT adversary $\mathcal{B}$ that breaks the $i\mathcal{O}$ security with the same probability. $\mathcal{B}$ runs Step 1 and Step 3 as in experiment 4. If the experiment does not abort, $\mathcal{B}$ chooses values to construct the circuits: $v_1 \in G_1, a \longleftarrow \mathbb{Z}_N, (e'_{1,0}, e'_{1,1}), \ldots, (e'_{n,0}, e'_{n,1}) \in \mathbb{Z}_N^2$. Set $e_{i,b} = e'_{i,b} \cdot a$, if $h(x^*)_i = b$, else $e_{i,b} = e'_{i,b}$. Let $sk' = (v_1, w, \gamma, (e_{1,0}, e_{1,1}), \ldots, (e_{n,0}, e_{n,1}))$, $\overrightarrow{V} = (v_1, v_1^a, \ldots, v_1^{a^{n-1}})$, and $sk'' = (v_1, w, \gamma, (e'_{1,0}, e'_{1,1}), \ldots, (e'_{n,0}, e'_{n,1}))$. $\mathcal{B}$ constructs circuits $C_0 = \mathcal{C}_3$ and $C_1 = \mathcal{C}_5$, where sk' is replaced by sk''. Then, he sends C_0 and C_1 to the $i\mathcal{O}$ challenger and gets $pk' = i\mathcal{O}(C_\beta)$. $\mathcal{B}$ computes $sk_{x^*} = i\mathcal{O}(\mathcal{C}_4), h(x^*) = b_1^*, \ldots, b_n^*$, $y_0 = e(v_1^{\prod_{j=1}^n e_{j,b_j^*}}, w^\gamma)$, and $y_1 \longleftarrow G_T$ and returns $(pk', sk_{x^*}, y_\alpha)$ to the adversary $\mathcal{A}$. $\mathcal{A}$ outputs α', if $\alpha = \alpha'$ and $\mathcal{B}$ outputs 0, else outputs 1. The circuits C_0 and C_1 have identical functionality. We observe that for any string x, $h(x) = b_1, \ldots, b_n$:

(i) For any $x \in \{0,1\}^\ell$ such that $H_u(x) = 1$, both circuits output $D(x) = e(v^{\prod_{j=1}^n d_{j,b_j}}, w)$

(ii) For any $x \in \{0,1\}^\ell$ such that $H_u(x) = 0, x \neq x^*$, we have $C_0(x) = (v_1^{\prod_{j=1}^n e_{j,b_j}}, w) = ((v_1^{a^{n(x)}})^{\prod_{j=1}^n e'_{j,b_j}}, w) = C_1(x)$, where $n(x) = |\{i : b_i = b_i^*\}|$

(iii) For any $x \in \{0,1\}^\ell$ such that $H_u(x) = 0, x = x^*$, we have $C_0(x) = (v_1^{\prod_{j=1}^n e_{j,b_j}}, w) = D_{x^*} = C_1(x)$

Therefore, if there exists an adversary $\mathcal{A}$ that distinguishes the outputs of Game 4 and Game 4_A with advantage ϵ, then there exists an adversary $\mathcal{B}$ that breaks the security of $i\mathcal{O}$ with the same advantage.

The indistinguishability of Game 4_A and Game 5 is similar to the previous one (Game 4 and Game 4_A). □

4.2.6. Game 6. This game is the same as the previous one except that $e(v_1^{a^n}, w^\gamma)$ is replaced by a random element from G_T. Formally, the challenger chooses a random element $T \longleftarrow G_T$, and uses $y_0 = T^{\prod_{j=1}^n e'_{j,b_j^*}}$ to replace $y_0 = e(v_1^{a^n \prod_{j=1}^n e'_{j,b_j^*}}, w^\gamma)$.

Lemma 5. *If there exists an adversary $\mathcal{A}$ that distinguishes Game 5 and Game 6 with advantage ϵ, then there exists an adversary $\mathcal{B}$ that breaks assumption 2 with advantage ϵ.*

Proof. We observe that the difference between Game 5 and Game 6 is that the element $e(v_1^{a^n}, w^\gamma)$ in Game 5 is replaced by a random element in Game 6. $\mathcal{B}$ receives an instance $(N, G_p, G_1, G_1, G_2, G_T, e, g_1, g_1^a, \ldots, g_1^{a^{n-1}}, g_2, g_2^\gamma, T)$, where T is either equal to $e(g_1^{a^n}, g_2^\gamma)$ or a random element of G_T. Then, $\mathcal{B}$ simulates Game 5 except that $y_\alpha = T^{\prod_{j=1}^n e'_{j,b_j^*}}$. $\mathcal{A}$ outputs α',. If $\alpha = \alpha'$, $\mathcal{B}$ outputs 0, which indicates that $T = e(g_1^{a^n}, g_2^\gamma)$; else, $\mathcal{B}$ outputs 1, which implies that T is a random element from G_T.

We observe that both y_0 and y_1 are chosen randomly from G_T in Game 6. This completes the proof of Theorem 1. □

4.3. Proof of Lemma 2. The major difference between Game 2 and Game 3 is the 'challenge partition' inputs x where $H_u(x) = 0$. Therefore, in order to show that for any PPT adversary $\mathcal{A}$, the outputs of Game 2 and Game 3 are indistinguishable; we give a sequence of subexperiments Game 2_A to Game 2_F and prove that any PPT attacker's advantage in each game must be negligible close to the previous one. We omit the previous experiment Game 2 and describe the intermediate experiments. In the first game, we change the secret key such that the circuit computes the output in a different manner and the output is the same as in the original circuit. Next, using the weak bilinear Diffie–Hellman inversion assumption, we modify the constants hardwired in

Input: a value x
Constants: v,w$^{\gamma}$, $(d_{1,0}, d_{1,1}), \ldots,(d_{n,0}, d_{n,1})$, h, the puncturable point $x^* \in \{0,1\}^{\ell(\lambda)}$, the vector $\boldsymbol{V} = (v_1, v_1^{a}, \ldots, v_1^{a^{n-1}})$, the secret key $sk' = (v_1, w, \gamma, (e'_{1,0}, e'_{1,1}), \ldots,(e'_{n,0}, e'_{n,1}))$, the random string $u \in \{0,1,\perp\}^n$, and the value Dx^*
(a) compute $h(x) = b_1, \ldots, b_n$, and $h(x^*) = b_1^*, \ldots, b_n^*$, let $n(x) = |\{i : b_i = b_i^*\}|$;
(b) if $H_u(x) = 0, x \neq x^*$, then output $D(x) = e((v_1^{a^{n(x)}})^{\Pi_{j=1}^n e_{j,b_j}}, w)$;
(c) if $H_u(x) = 0, x = x^*$, then output Dx^*;
(d) if $H_u(x) = 1$, output $D(x) = e(v^{\Pi_{j=1}^n d_{j,b_j}}, w)$.

Figure 5: Circuit $\mathscr{C}_5$.

Input: a value x
Constants: the secret key sk, the puncturable point $x^* \in \{0,1\}^{\ell(\lambda)}$, the secret key $sk' = (v_1, w, \gamma, (e_{1,0}, e_{1,1}), \ldots,(e_{n,0}, e_{n,1}))$, the vector $\boldsymbol{V} = (v_1, v_1^{a}, \ldots, v_1^{a^{n-1}})$, and the random string $u \in \{0,1,\perp\}^n$
(a) compute $h(x) = b_1, \ldots, b_n$, and $h(x^*) = b_1^*, \ldots, b_n^*$, let $n(x) = |\{i : b_i = b_i^*\}|$;
(b) if $x = x^*$, output$(\perp, \perp)$;
(c) else if $H_u(x) = 0$, then output $y = e((v_1^{a^{n(x)}})^{\Pi_{j=1}^n d_{j,b_j}}, w^{\gamma})$, $\pi = (v_1^{a^{n(x)}})^{\Pi_{j=1}^n d_{j,b_j}}$;
(d) if $H_u(x) = 1$,output $y = e(v^{\Pi_{j=1}^n d_{j,b_j}}, w^{\gamma})$, $\pi = v^{\Pi_{j=1}^n d_{j,b_j}}$.

Figure 6: Circuit $\mathscr{C}_6$.

the program such that the output of all challenge partition inputs is changed. Essentially, a different base for the challenge partition is used in the two programs, respectively. Finally, using Subgroup Hiding Assumption and Chinese Remainder Theorem, we can change the exponents for the challenge partition and ensure that the original circuit (in Game 2) and final circuit (in Game 3) use different secret keys for the challenge partition.

4.3.1. Game 2_A. In this game, we modify the secret key $d_{i,b}$ for $j \in \{1, \ldots, n\}$ and $b \in \{0,1\}$. It is easy to verify that the two experiments are statistically indistinguishable. The detailed description is given as follows:

(1) The challenger flips a coin $\alpha \longleftarrow \{0,1\}$ and runs $u \longleftarrow \mathrm{AdmSample}(1^{\lambda}, Q)$. Then, the challenger runs $(N, G_p, G_q, G_1, G_2, G_T, e) \longleftarrow \mathscr{G}(1^{\lambda})$, chooses $v \in G_1$, $w \in G_2, a, \gamma \in \mathbb{Z}_N$, and $(d'_{1,0}, d'_{1,1}), \ldots, (d'_{n,0}, d'_{n,1}) \in \mathbb{Z}_N^2$, sets $d_{i,b} = d'_{i,b}$ if $u_i = b$, else $d_{i,b} = a \cdot d'_{i,b}$, and $sk = (v, w, \gamma, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$ and $pk = (w, w^{\gamma}, i\mathcal{O}(\mathscr{C}_1))$.

(2) The adversary $\mathscr{A}$ makes an evaluation query $x_i \in \{0,1\}^{\ell(\lambda)}$. The challenger checks if $H_u(x_i) = 1$ (recall that $H_u(x) = 0$, if $h(x)_j \neq u_j$ for all $j \in [n]$). If not, the game aborts. Else, the challenger computes $h(x_i) = b_1^i, \ldots, b_n^i$ and outputs $(F(sk, x_i), P(sk, x_i)) = (e(v^{\prod_{j=1}^n d_{j,b_j^i}}, w^{\gamma}), v^{\prod_{j=1}^n d_{j,b_j^i}})$.

(3) The adversary $\mathscr{A}$ sends a challenge point x^* such that $x^* \neq x_i$ for all $i \in [Q(\lambda)]$.

(4) The challenger checks if $H_u(x^*) = 0$. If not, the game aborts. Else, the challenger computes $sk_{x^*} \longleftarrow i\mathcal{O}(\mathscr{C}_2)$ and $h(x^*) = b_1^*, \ldots, b_n^*$, sets $y_0 = e(v^{\prod_{j=1}^n d_{j,b_j^*}}, w^{\gamma})$ and $y_1 \longleftarrow G_T$, and returns $(pk, sk_{x^*}, y_{\alpha})$ to the adversary $\mathscr{A}$.

(5) The adversary $\mathscr{A}$ outputs a bit α' and wins if $\alpha' = \alpha$.

Lemma 6. *The outputs of Game 2 and Game* 2_A *are statistically indistinguishable.*

Proof. We observe the difference between Game 2 and Game 2_A is the manner in which $d_{i,b}$ are chosen. In Game 2, $d_{i,b}$ are chosen randomly from $\mathbb{Z}_N$, while in Game 2_A, the challenger first chooses $d'_{i,b} \longleftarrow \mathbb{Z}_N$ and $a \longleftarrow \mathbb{Z}_N$ and sets $d_{i,b} = d'_{i,b}$, if $u_i = b$, else $d_{i,b} = d'_{i,b} \cdot a$. Since $a \in \mathbb{Z}_N$, which is invertible with overwhelming probability, $d_{i,b} = d'_{i,b} \cdot a$ is a uniformly random element in $\mathbb{Z}_N$. Therefore, the two experiments are statistically indistinguishable. □

4.3.2. Game 2_B. This game is the same as the previous one except the hardwire of the circuit is changed. The domain is divided into two disjoint sets by the admissible hash function. When $H_u(x) = 0$, all elements $d_{i,b}$ used to compute function values y contain a factor a. Therefore, the related function values can be computed by $v' = v^{a^n}$. On the other hand, only some elements $d_{i,b}$ used to compute function values y contain the factor a when $H_u(x) = 1$. Therefore, the related function values can only be computed by $(v, v^a, \ldots, v^{a^{n-1}})$.

(1) The challenger flips a coin $\alpha \longleftarrow \{0,1\}$ and runs $u \longleftarrow \mathrm{AdmSample}(1^{\lambda}, Q)$. Let $m(x) = |\{i : u_i \neq h(x)_i\}|$. Then, the challenger runs $(N, G_p, G_q, G_1, G_2, G_T, e) \longleftarrow \mathscr{G}(1^{\lambda})$, chooses $v \in G_1, w \in G_2, a, \gamma \in \mathbb{Z}_N$, and $(d'_{1,0}, d'_{1,1}), \ldots, (d'_{n,0}, d'_{n,1}) \in \mathbb{Z}_N^2$, sets $d_{i,b} = d'_{i,b}$ if $u_i = b$, else $d_{i,b} = a \cdot d'_{i,b}$, and $D = ((d'_{1,0}, d'_{1,1}), \ldots, (d'_{n,0}, d'_{n,1}))$, $\overrightarrow{V} = (v, v^a, \ldots, v^{a^{n-1}})$, $v' = v^{a^n}$, and $pk = (w, w^{\gamma}, i\mathcal{O}(\mathscr{C}_7))$, where the description of $\mathscr{C}_7$ is given in Figure 7.

(2) The adversary $\mathscr{A}$ makes an evaluation query $x_i \in \{0,1\}^{\ell(\lambda)}$. The challenger checks if $H_u(x_i) = 1$ (recall that $H_u(x) = 0$, if $h(x)_j \neq u_j$ for all $j \in [n]$). If not, the game aborts. Else, the challenger computes

Input: a value x
Constants: the element $w \in G_2$, $v' \in G_1$, $D = ((d'_{1,0}, d'_{1,1}), \ldots, (d'_{n,0}, d'_{n,1}))$, the random string $u \in \{0, 1, \perp\}^n$, and $\boldsymbol{V} = (v, v^a, \ldots, v^{a^{n-1}})$
(a) compute $h(x) = b_1, \ldots, b_n$, and $m(x) = |\{i : u_i \neq h(x)_i\}|$;
(b) if $H_u(x) = 0$, output $D(x) = e((v')^{\Pi_{j=1}^n d_{j,b_j}}, w)$;
(c) if $H_u(x) = 1$, output $D(x) = e((v^{a^{m(x)}})^{\Pi_{j=1}^n d_{j,b_j}}, w)$.

Figure 7: Circuit $\mathscr{C}_7$.

$h(x_i) = b_1^i, \ldots, b_n^i$ and outputs $(F(sk, x_i), P(sk, x_i)) = (e(v^{\prod_{j=1}^n d_{j,b_j^i}}, w^\gamma), v^{\prod_{j=1}^n d_{j,b_j^i}})$.

(3) The adversary $\mathscr{A}$ sends a challenge point x^* such that $x^* \neq x_i$ for all $i \in [Q(\lambda)]$.

(4) The challenger checks if $H_u(x^*) = 0$. If not, the game aborts. Else, the challenger computes $sk_{x^*} \longleftarrow i\mathcal{O}(\mathscr{C}_8)$ and $h(x^*) = b_1^*, \ldots, b_n^*$, sets $y_0 = e(v^{\prod_{j=1}^n d_{j,b_j^*}}, w^\gamma)$ and $y_1 \longleftarrow G_T$, and returns (pk, sk_{x^*}, y_α) to the adversary $\mathscr{A}$, where the description of $\mathscr{C}_8$ is given in Figure 8.

(5) The adversary $\mathscr{A}$ outputs a bit α' and wins if $\alpha' = \alpha$.

Lemma 7. *Assuming $i\mathcal{O}$ is a secure indistinguishability obfuscator, Game 2_A and Game 2_B are computationally indistinguishable.*

Proof. The proof method is similar to Lemma 4. □

4.3.3. Game 2_C. This game is the same as the previous one except that v' is chosen randomly from G_1 in Step 1, and $y_0 = e((v')^{\prod_{j=1}^n d'_{j,b_j^*}}, w^\gamma)$ in Step 4.

Lemma 8. *If there exists an adversary $\mathscr{A}$ that distinguishes the Game 2_B and Game 2_C with advantage ϵ, then there exists an adversary $\mathscr{B}$ that breaks assumption 2 with advantage ϵ.*

Proof. We observe that the difference between Game 2_B and Game 2_C is that the term v^{a^n} is replaced by a random element of G_1. This proof is similar to the proof of Lemma 5. □

4.3.4. Game 2_D. This game is the same as the previous one except that v is chosen randomly from the subgroup G_p, and $v\prime$ is chosen randomly from the subgroup G_q in Step 1.

Lemma 9. *Assuming assumption 1 holds, Game 2_C and Game 2_D are computationally indistinguishable.*

Proof. We introduce an intermediate experiment 2_{C_1} and show that Game 2_{C_1} and 2_C are computationally indistinguishable. Similarly, Game 2_{C_1} and Game 2_D are computationally indistinguishable.

Game 2_{C_1} is the same as Game 2_C except that v is chosen from G_p. Suppose that there exists an adversary $\mathscr{A}$ which can distinguish Game 2_{C_1} and 2_C, we construct an adversary $\mathscr{B}$ that breaks assumption 1. $\mathscr{B}$ receives $(N, G_p, G_q, G_1, G_2, G_T, e, T)$, where $T \longleftarrow G_1$ or $T \longleftarrow G_p$. $\mathscr{B}$ sets $v = T$, chooses $v' \longleftarrow G_1, w \longleftarrow G_2, a, \gamma \xleftarrow{} \mathbb{Z}_N$, and $(d'_{1,0}, d'_{1,1}), \ldots, (d'_{n,0}, d'_{n,1}) \in \mathbb{Z}_N^2$, and computes V, D, and pk as in Game 2_C. Then, $\mathscr{B}$ runs the rest steps as in Game 2_C. At last, $\mathscr{A}$ outputs α', if $\alpha = \alpha'$ and $\mathscr{B}$ guesses $T \in G_1$, else $\mathscr{B}$ guesses $T \in G_p$. Note that $\mathscr{B}$ simulates exactly Game 2_C when $T \in G_1$, and $\mathscr{B}$ simulates exactly Game 2_{C_1} when $T \in G_p$. Therefore, if there exists an adversary $\mathscr{A}$ that distinguishes the outputs 2_C and 2_{C_1} with advantage ϵ, then there exists an adversary $\mathscr{B}$ that breaks the assumption 1. □

4.3.5. Game 2_E. This game is the same as the previous one except that the secret key is divided into two parts sk and sk'. If $H_u(x) = 0$, the related function values are computed by sk'. Else, the related function values are computed by sk.

(1) The challenger flips a coin $\alpha \longleftarrow \{0, 1\}$ and runs $u \longleftarrow \mathrm{AdmSample}(1^\lambda, Q)$. Let $m(x) = |\{i : u_i \neq h(x)_i\}|$. Then, the challenger runs $(N, G_p, G_q, G_1, G_2, G_T, e) \longleftarrow \mathscr{G}(1^\lambda)$, chooses $v \in G_p, v' \in G_q, w \in G_2, a, \gamma \in \mathbb{Z}_N$, and $(d_{1,0}, d_{1,1}, \ldots, (d_{n,0}, d_{n,1})) \in \mathbb{Z}_N^2$, and sets $sk = (v, w^\gamma, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$, $sk' = (v', w^\gamma, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$, and $pk = (w, w^\gamma, i\mathcal{O}(\mathscr{C}_3))$.

(2) The adversary $\mathscr{A}$ makes an evaluation query $x_i \in \{0, 1\}^{\ell(\lambda)}$. The challenger checks if $H_u(x_i) = 1$ (recall that $H_u(x) = 0$ if $h(x)_j \neq u_j$ for all $j \in [n]$). If not, the game aborts. Else, the challenger computes $h(x_i) = b_1^i, \ldots, b_n^i$ and outputs $(F(sk, x_i), P(sk, x_i)) = (e(v^{\prod_{j=1}^n d_{j,b_j^i}}, w^\gamma), v^{\prod_{j=1}^n d_{j,b_j^i}})$.

(3) The adversary $\mathscr{A}$ sends a challenge point x^* such that $x^* \neq x_i$ for all $i \in [Q(\lambda)]$.

(4) The challenger checks if $H_u(x^*) = 0$. If not, the game aborts. Else, the challenger computes $sk_{x^*} \longleftarrow i\mathcal{O}(\mathscr{C}_4)$ and $h(x^*) = b_1^*, \ldots, b_n^*$, sets $y_0 = e((v')^{\prod_{j=1}^n d'_{j,b_j^*}}, w^\gamma)$ and $y_1 \longleftarrow G_T$, and returns (pk, sk_{x^*}, y_α) to the adversary $\mathscr{A}$.

(5) The adversary $\mathscr{A}$ outputs a bit α' and wins if $\alpha' = \alpha$.

Lemma 10. *Assuming $i\mathcal{O}$ is a secure indistinguishability obfuscator, Game 2_D and Game 2_E are computationally indistinguishable.*

Input: a value x
Constants: the puncturable value x^*, the element $w^\gamma \in G_2, v' G_t$, $D = ((d'_{1,0}, d'_{1,1}), \cdots, (d'_{n,0}, d'_{n,1}))$, the random string $u \in \{0, 1, \perp\}^n$, and $\boldsymbol{V} = (v, v^a, \ldots, v^{a^{n-1}})$
(a) compute $h(x) = b_1, \ldots, b_n$, and $m(x) = |\{i : u_i \neq h(x)_i\}|$;
(b) if $x = x^*$, output $(\perp, \perp)$;
(c) else if $H_u(x) = 0$, output $y = e((v')^{\Pi_{j=1}^n d'_{j,b_j}}, w^\gamma)$, $\pi = (v')^{\Pi_{j=1}^n d'_{j,b_j}}$;
(d) if $H_u(x) = 1$, output $D(x) = e((v^{a^{m(x)}})^{\Pi_{j=1}^n d'_{j,b_j}}, w^\gamma)$, $\pi = (v^{a^{m(x)}})^{\Pi_{j=1}^n d'_{j,b_j}}, w^\gamma$

Figure 8: Circuit $\mathscr{C}_8$.

Proof. We introduce two intermediate experiments 2_{D_1} and 2_{D_2} and show that Game 2_D and 2_{D_1} are computationally indistinguishable, Game 2_{D_1} and Game 2_{D_2} are computationally indistinguishable, and Game 2_{D_2} and Game 2_E are computationally indistinguishable.

First, we define the experiments 2_{D_1} and 2_{D_2}. In experiment 2_{D_1}, the challenger samples v, v', w, a, γ, and $d'_{i,b}$ as in Game 2_D, sets $d_{i,b} = d'_{i,b}$ if $u_i = b$, else $d_{i,b} = a \cdot d'_{i,b}$. Then, it answers the evaluation queries of $\mathscr{A}$ exactly as in Game 2_D. For the challenge queries, it sets $v'' = (v')^{1/a^n}$ and computes $y_0 = e((v'')^{\prod_{j=1}^n d_{j,b_j^*}}, w^\gamma)$ and $y_1 \longleftarrow G_T$. The public key pk is computed by the circuit $\mathscr{C}_3$, where $sk = (v, w^\gamma, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$ and $sk' = (v'', w^\gamma, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$. The constrained key sk_{x^*} is computed as in Game 2_D.

Game 2_{D_2} is the same as the game 2_{D_1}, except that the constrained key sk_{x^*} is computed by the circuit $\mathscr{C}_4$. □

Claim 1. Assuming $\mathscr{A}$ is a secure indistinguishability obfuscator, Game α' and Game $\alpha' = \alpha$ are computationally indistinguishable.

Proof. We construct a PPT adversary $i\mathcal{O}$ that uses 2_A to break the security of 2_B. 2_C runs Step 1 and Step 3 as in Game v'. On receiving the challenge point G_1, $y_0 = e((v')^{\prod_{j=1}^n d'_{j,b_j^*}}, w^\gamma)$ sets $\mathscr{A}$ and 2_B as in 2_C and constructs circuits $\mathscr{B}$ and 2_B. Then, he sends 2_C to the v^{a^n} challenger and receives G_1. 2_D computes G_p as in Game v' and sends G_q to 2_C. 2_D returns 2_{C_1}, if 2_{C_1}, 2_C outputs 0, else outputs 1.

Next, we only show that the circuit 2_{C_1} and 2_D have the identical functionality. For any 2_{C_1} such that 2_C. For any G_p such that $\mathscr{A}$. Therefore, the two circuits are functionally equivalent. Hence, if there exists an adversary that can distinguish the two games, then we can construct an adversary 2_{C_1} that breaks the 2_C security. □

Claim 2. Assuming $\mathscr{B}$ is a secure indistinguishability obfuscator, Game $\mathscr{B}$ and Game $(N, G_p, G_q, G_1, G_2, G_T, e, T)$ are computationally indistinguishable.

Proof. The proof method is similar to the previous one. □

Claim 3. Game $T \longleftarrow G_1$ and Game $T \longleftarrow G_p$ are statistically indistinguishable.

Proof. The difference between Game $\mathscr{B}$ and Game $v = T$ is the chosen way of $v' \longleftarrow G_1, w \longleftarrow G_2, a, \gamma \in \mathbb{Z}_N$. In addition, $(d'_{1,0}, d'_{1,1}), \ldots, (d'_{n,0}, d'_{n,}) \in \mathbb{Z}_N^2$ is replaced by the value $\overrightarrow{V}$, D, and pk in Game 2_C. Since $\mathscr{B}$, a is invertible with overwhelming probability. Therefore, 2_C is a uniform element from $\mathscr{A}$ and α' is also a uniformly random element from $\alpha = \alpha'$ in Game $\mathscr{B}$. It follows that the two experiments are statistically indistinguishable. □

4.3.6. Game 2_F. This game is the same as the previous one except that the generation way of secret key sk' is different to the one of sk.

(1) The challenger flips a coin $\alpha \longleftarrow \{0, 1\}$ and runs $u \longleftarrow \mathrm{AdmSample}(1^\lambda, Q)$. Let $m(x) = |\{i : u_i \neq h(x)_i\}|$. Then, the challenger runs $(N, G_p, G_q, G_1, G_2, G_T, e) \longleftarrow \mathscr{G}(1^\lambda)$, chooses $v \in G_p, v' \in G_q, w \in G_2, a, \gamma \in \mathbb{Z}_N$, $(d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}) \in \mathbb{Z}_N^2$, and $(e_{1,0}, e_{1,1}), \ldots, (e_{n,0}, e_{n,1}) \in \mathbb{Z}_N^2$, and sets $sk = (v, w^\gamma, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$, $sk' = (v', w^\gamma, (e_{1,0}, e_{1,1}), \ldots, (e_{n,0}, e_{n,1}))$, and $pk = (w, w^\gamma, i\mathcal{O}(\mathscr{C}_3))$.

(2) The adversary $\mathscr{A}$ makes an evaluation query $x_i \in \{0, 1\}^{\ell(\lambda)}$. The challenger checks if $H_u(x_i) = 1$ (recall that $H_u(x) = 0$ if $h(x)_j \neq u_j$ for all $j \in [n]$). If not, the game aborts. Else, the challenger computes $h(x_i) = b_1^i, \ldots, b_n^i$ and outputs $(F(sk, x_i), P(sk, x_i)) = (e(v^{\prod_{j=1}^n d_{j,b_j^i}}, w^\gamma), v^{\prod_{j=1}^n d_{j,b_j^i}})$.

(3) The adversary $\mathscr{A}$ sends a challenge point x^* such that $x^* \neq x_i$ for all $i \in [Q(\lambda)]$.

(4) The challenger checks if $H_u(x^*) = 0$. If not, the game aborts. Else, the challenger computes $sk_{x^*} \longleftarrow i\mathcal{O}(\mathscr{C}_4)$ and $h(x^*) = b_1^*, \ldots, b_n^*$, sets $y_0 = e((v')^{\prod_{j=1}^n d'_{j,b_j^*}}, w^\gamma)$ and $y_1 \longleftarrow G_T$, and returns (pk, sk_{x^*}, y_α) to the adversary $\mathscr{A}$.

(5) The adversary $\mathscr{A}$ outputs a bit $\alpha\prime$ and wins if $\alpha' = \alpha$.

Lemma 11. *Game 2_E and Game 2_F are statistically indistinguishable.*

Proof. The only difference between Game 2_E and Game 2_F is the chosen way of the secret key sk and sk'. In Game 2_E, the challenger chooses $v \in G_p, v' \in G_q, w \in G_2, \gamma, d_{i,b} \in \mathbb{Z}_N$ and sets $sk = (v, w^\gamma, (d_{1,0}, d_{1,1})), \ldots, (d_{n,0}, d_{n,1}))$ and $sk' = (v', w^\gamma, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$. While the challenger chooses $d_{i,b}, e_{i,b} \in \mathbb{Z}_N$ and sets $sk = (v, w^\gamma, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$ and $sk' = (v', w^\gamma, (e_{1,0}, e_{1,1}), \ldots, (e_{n,0}, e_{n,1}))$ in Game 2_F. Using Chinese remainder theorem, the

distributions $\{x \bmod p, x \bmod q : x \longleftarrow \mathbb{Z}_N\}$ and $\{x \bmod p, y \bmod q : x, y \longleftarrow \mathbb{Z}_N\}$ are statistically indistinguishable. Therefore, Game 2_E and Game 2_F are statistically indistinguishable. □

Lemma 12. *Assuming assumption 1 holds, Game 2_F and Game 3 are computationally indistinguishable.*

Proof. The proof method is similar to Lemma 9. □

5. Constrained Verifiable Random Function

In this section, we give our construction of constrained verifiable random function with polynomial size of the constrained set. We embed the puncturable VRFs in the constrained VRFs. Informally, our algorithm works as follows: The setup algorithm is the same as the puncturable VRFs. The constrained key sk_S for the subset S is a circuit which has the secret key sk hardwired in it. On input a value x, the circuit computes the function value and proof by the puncturable VRFs if $x \notin S$. The verifiable algorithm is the same as the puncturable VRFs. When proving the pseudorandomness, we translate puncturable VRFs into constrained VRFs with polynomial size of the constrained set by means of hybrid argument. Once the adversary queries the constrained key for the polynomial set S_1, the challenger can guess the challenge point x^* with a probability of $1/|S_1|$. Subsequently, the secret key sk can be replaced by a constrained key sk_{x^*} of puncturable VRFs. Via a hybrid argument, we reduce pseudorandomness of puncturable VRFs to the pseudorandomness of constrained VRFs.

Let $F : \mathcal{K} \times \mathcal{X} \longrightarrow \mathcal{Y}$ be a puncturable VRF (Setup, Puncture, Prove, and Verify), and $P : \mathcal{K} \times \mathcal{X} \longrightarrow G_1$ be a function of generation proof. We construct constrained VRFs (F.Setup, F.Constrain, F.Prove, and F.Verify) by invoking the puncturable VRFs:

(i) $F.\text{Setup}(1^\lambda) \longrightarrow (pk, sk)$: Run the algorithm $(pk_1, sk_1) \longleftarrow \text{Setup}(1^\lambda)$. Set $pk = pk_1$ and $sk = sk_1$.

(ii) $F.\text{Constrain}(sk, S) \longrightarrow sk_S$: This algorithm takes the secret key sk and the constrained set S as inputs, where $|S|$ = poly and computes an obfuscation of a circuit $\mathcal{C}_{sk,S}$ defined as in Figure 9. $\mathcal{C}_{sk,S}$ has the secret key, the function descriptions F and P, and the constrained set S hardwired in it. Sets $sk_S \longleftarrow i\mathcal{O}(C_{sk}, S)$ where $\mathcal{C}_{sk,S}$ is padded to be of appropriate size.

(iii) $F.\text{Prove}(sk_S, x) \longrightarrow (y, \pi)$ or $(\perp, \perp)$: The constrained key sk_S is a program that takes x as the input. Define $F.\text{Prove}(sk_S, x) = sk_S(x)$.

(iv) $F.\text{Verify}(pk, x, y, \pi) \longrightarrow \{0, 1\}$: This algorithm is the same as Verify.

The provability and uniqueness follow from the puncturable VRFs. We omit the detailed description. Next, we show that this construction satisfies the pseudorandomness defined in Section 3.

Theorem 2. *Assuming $i\mathcal{O}$ is a secure indistinguishability obfuscator and (Setup, Puncture, Prove, and Verify) is a secure puncturable VRF. Then, the construction defined above satisfies the pseudorandomness.*

Proof. Without loss of generality, we assume the adversary makes q_1 evaluation queries and q_2 constrained queries. We present a full description of each game and underline the changes from the presented one to the previous one. □

5.1. Game 1. The first game is the original security game for our construction. Here, the challenger first chooses a pair constrained VRF key (pk, sk). Then, $\mathcal{A}$ makes evaluation queries and constrained key queries and outputs a challenge point. The challenger responds with either a VRF evaluation or a random element.

(i) The challenger chooses $b \longleftarrow \{0, 1\}$ and then generates (pk, sk) by running the algorithm F.Setup (1^λ)

(ii) The adversary makes evaluation queries or constrained queries:

(1) If $\mathcal{A}$ sends an evaluation query x_i, then output $(F(sk, x_i), P(sk, x_i))$

(2) If $\mathcal{A}$ sends a constrained key query for S_j, output the constrained key $sk_{S_j} \longleftarrow i\mathcal{O}(\mathcal{C}_{sk,S_j})$

(iii) $\mathcal{A}$ sends a challenge query x^* such that $x^* \neq x_i$ for all $i \leq q_1$ and $x^* \in S_j$ for all $j \leq q_2$. Then, the challenger sets $y_0 = F(sk, x^*)$ and $y_1 \longleftarrow \mathcal{Y}$ and outputs (y_b, pk)

(iv) $\mathcal{A}$ outputs b' and wins if $b = b'$

5.2. Game 2. This game is the same as the previous one except that we introduce an abort condition. When the adversary $\mathcal{A}$ makes the first constrained query S_1, the challenger guesses a challenge query $x' \in S_1$. If the last $q_2 - 1$ queries S_j does not contain x', the experiment aborts. In addition, the experiment aborts if $x' \neq x^*$, where x^* is the challenge query.

(i) The challenger chooses $b \longleftarrow \{0, 1\}$ and then generates (pk, sk) by running the algorithm F.Setup (1^λ).

(ii) The adversary makes evaluation queries or constrained queries: For the first constrained query S_1, the challenger chooses $x' \longleftarrow S_1$ and output $sk_{S_1} \longleftarrow i\mathcal{O}(\mathcal{C}_{sk,S_1})$. For all evaluation queries x_i before the first constrained query, the challenger outputs $(F(sk, x_i), P(sk, x_i))$. For all queries after the first constrained query, the challenger does as follows:

(1) If $\mathcal{A}$ sends an evaluation query x_i such that $x_i = x'$, the experiment aborts. Else, if $x_i \neq x'$, output $(F(sk, x_i), P(sk, x_i))$

Input: a value $x \in X$
Constants: the function description F and P, the secret key sk, the constrained set $S \in X$

1. if $S \in X$, output $(\perp, \perp)$;
2. else, output $y = F(sk, x)$, $\pi = P(sk, x)$.

Figure 9: Circuit $\mathscr{C}_{sk,S}$.

(2) If $\mathscr{A}$ sends a constrained key query for S_j such that $x_i \notin S_j$, the experiment aborts. Else, output $sk_{S_j} \longleftarrow i\mathcal{O}(\mathscr{C}_{sk,S_j})$.

(iii) $\mathscr{A}$ sends a challenge query x^* such that $x^* \neq x_i$ for all $i \leq q_1$, and $x^* \in S_j$ for all $j \leq q_2$. If $x^* \neq x'$, the experiment aborts. Else, the challenger sets $y_0 = F(sk, x^*)$ and $y_1 \longleftarrow \mathscr{Y}$ and outputs (y_b, pk).

(iv) $\mathscr{A}$ outputs b' and wins if $b = b'$.

Lemma 13. *For any PPT adversary $\mathscr{A}$, if $\mathscr{A}$ wins with advantage ϵ in Game 1, then it wins with advantage $\varepsilon/|S_1|$ in Game 2.*

Proof. According the pseudorandomness defined in Section 3, the challenge point belongs to the constrained set. The two experiments are equal if Game 2 does not abort. Since the challenger guesses correctly with probability $1/|S_1|$, if $\mathscr{A}$ wins with advantage ϵ in Game 1, then it wins with advantage $\varepsilon/|S_1|$ in Game 2. □

5.3. Game 2_i. For $0 \leq i \leq q_2$, the experiment is the same as the previous one except that the constrained queries use $sk_{x'}$ instead of sk in the first i experiment. We observe that the Game 2_0 is equal to the Game 2.

(i) The challenger chooses $b \longleftarrow \{0,1\}$ and then generates (pk, sk) by running the algorithm F.Setup (1^λ).

(ii) The adversary makes evaluation queries or constrained queries: For the first constrained query S_1. The challenger chooses $x' \longleftarrow S_1$, computes $sk_{x'} \longleftarrow i\mathcal{O}(\mathscr{C}_{sk_{x'},S_1})$ and $\pi\prime = \text{Puncture}(sk, x')$, and outputs $sk_{S_1} \longleftarrow i\mathcal{O}(\mathscr{C}_{sk_{x'},S_1})$, where the description of the circuit $\mathscr{C}_{sk_{x'},S_1}$ is given in Figure 10. For all evaluation queries x_i before the first constrained query, the challenger outputs $(F(sk, x_i), P(sk, x_i))$. For all queries after the first constrained query, the challenger does as follows:

(1) If $\mathscr{A}$ sends an evaluation query x_i such that $x_i = x'$, the experiment aborts. Else, if $x_i \neq x'$, output $(F(sk, x_i), P(sk, x_i)) = \text{Prove}(sk_{x'}, x_i)$

(2) If $\mathscr{A}$ sends a constrained key query for S_j such that $x_i \notin S_j$, the experiment aborts. Else, if $j \leq i$ output $sk_{S_j} \longleftarrow i\mathcal{O}(\mathscr{C}_{sk_{x'},S_j})$, else output $sk_{S_j} \longleftarrow i\mathcal{O}(\mathscr{C}_{sk,S_j})$, where the description of the circuit $\mathscr{C}_{sk_{x'},S_j}$ is given in Figure 10.

(iii) $\mathscr{A}$ sends a challenge query x^* such that $x^* \neq x_i$ for all $i \leq q_1$ and $x^* \in S_j$ for all $j \leq q_2$. If $x^* \neq x'$, the experiment aborts. Else, the challenger sets $y_0 = F(sk, x^*)$ and $y_1 \longleftarrow \mathscr{Y}$, computes, and outputs (y_b, pk).

(iv) $\mathscr{A}$ outputs b' and wins if $b = b'$.

Lemma 14. *Assuming $i\mathcal{O}$ is a secure indistinguishability obfuscator, Game 2_{i-1} and 2_i are computationally indistinguishable.*

Proof. We observe that the difference between Game 2_{i-1} and 2_i respond to the $i\prime$th constrained query. In Game 2_{i-1}, $sk_{S_i} \longleftarrow i\mathcal{O}(\mathscr{C}_{sk,S_i})$, while in Game 2_i, $sk_{S_i} \longleftarrow i\mathcal{O}(\mathscr{C}_{sk_{x'},S_i})$. In order to prove that the two games are indistinguishable, we only need to show that the circuit $\mathscr{C}_{sk,S_i}$ and $\mathscr{C}_{sk_{x'},S_i}$ are functionally identical.

(i) If $x \in S_i$, both circuits output $(\perp, \perp)$

(ii) For any input $x \notin S_i$, $\mathscr{C}_{sk,S_i}(x) = \text{Prove}(sk, x) = \text{Prove}(sk_{x',S_i}, x) = \mathscr{C}_{sk_{x'},S_i}(x)$

Therefore, by the security of $i\mathcal{O}$, the two experiments are indistinguishable. □

5.4. Game 3. This game is the same as game 2_{q_2} except that y_0 is replaced by a random element from $\mathscr{Y}$.

(i) The challenger chooses $b \longleftarrow \{0,1\}$ and then generates (pk, sk) by running the algorithm F.Setup (1^λ).

(ii) The adversary makes evaluation queries or constrained queries: For the first constrained query S_1, the challenger chooses $x' \longleftarrow S_1$ and output $sk_{S_1} \longleftarrow i\mathcal{O}(\mathscr{C}_{sk,S_1})$. For all evaluation queries x_i before the first constrained query, the challenger outputs $(F(sk, x_i), P(sk, x_i))$. For all queries after the first constrained query, the challenger does as follows:

(1) If $\mathscr{A}$ sends an evaluation query x_i such that $x_i = x'$, the experiment aborts. Else, if $x_i \neq x'$, output $(F(sk, x_i), P(sk, x_i))$

(2) If $\mathscr{A}$ sends a constrained key query for S_j such that $x_i \notin S_j$, the experiment aborts. Else, output $sk_{S_j} \longleftarrow i\mathcal{O}(\mathscr{C}_{sk,S_j})$

(iii) $\mathscr{A}$ sends a challenge query x^* such that $x^* \neq x_i$ for all $i \leq q_1$ and $x^* \in S_j$ for all $j \leq q_2$. If $x^* \neq x'$, the experiment aborts. Else, the challenger sets $y_0 \longleftarrow \mathscr{Y}$ and $y_1 \longleftarrow \mathscr{Y}$ and outputs (y_b, pk).

(iv) $\mathscr{A}$ outputs b' and wins if $b = b'$.

Input: a value $x \in X$
Constants: the function description F, the secret key $sk_{x'}$, the constrained set $S_j \in X$
1. if $x \in S_j$, output $(\perp, \perp)$;
2. else, output $(y, \pi) = \text{Prove}(sk_{x'}, x)$.

Figure 10: Circuit $\mathscr{C}_{sk_{x'}, S_j}$.

Lemma 15. *Assuming the puncturable VRFs are secure, Game* 2_{q_2} *and Game 3 are computationally indistinguishable.*

Proof. We prove that if there exists an adversary $\mathscr{A}$ that distinguishes the Game 2_{q_2} and Game 3, then there exists another adversary $\mathscr{B}$ that breaks the security of puncturable VRFs.

$\mathscr{B}$ can simulate perfect experiment for $\mathscr{A}$. For each evaluation query x before the first constrained key query, $\mathscr{B}$ sends x to the puncturable VRFs' challenger and returns (y, π) to $\mathscr{A}$. When $\mathscr{A}$ queries the constrained key S_1, $\mathscr{B}$ chooses $x' \in S_1$, sends x' to the challenger, and receives $(sk_{x'}, pk, \text{and } y)$. Then, $\mathscr{B}$ uses $sk_{x'}$ to respond the remaining queries. On receiving the challenge input x^*, $\mathscr{B}$ checks $x' = x^*$ and outputs y. $\mathscr{B}$ outputs the response of $\mathscr{A}$. We observe that if y is chosen randomly, then $\mathscr{B}$ simulates Game 3, else it simulates Game 2_{q_2}. Therefore, Game 2_{q_2} and Game 3 are computationally indistinguishable.

We observe that both y_0 and y_1 are chosen randomly from $\mathscr{Y}$. Therefore, for any PPT adversary $\mathscr{A}$, it has negligible advantage in Game 3. This completes the proof of Theorem 2. □

6. Conclusion

In this work, we construct a novel constrained VRF for polynomial size set and give the proof of security under a new secure definition which is called semiadaptive security. Meanwhile, our construction is based on bilinear maps, which avoid the application of multilinear maps. Although it does not satisfy full adaptive security, it has solved some problems compared with selective security, which allows the adversary to query the evaluation oracle before it outputs the challenge point. To construct a fully adaptive security constrained VRFs is our future work.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (nos. 61871430 and 61971458), Science and Technology Project of Henan Educational Department (no. 20A520012), and Special Project of Key Scientific Research Projects of Henan Higher Education Institutions (no. 19zx010).

References

[1] O. Goldreich, S. Goldwasser, and S. Micali, "How to construct random functions," in *Proceedings of the 25th IEEE Symposium on Foundations of Computer Science*, pp. 464–479, Singer Island, FL, USA, October 1984.

[2] D. Boneh and B. Waters, "Constrained pseudorandom functions and their applications," in *Advances in Cryptology—ASIACRYPT 2013—Proceedings of the 19th International Conference on the Theory and Application of Cryptology and Information Security*, pp. 280–300, Bengaluru, India, December 2013.

[3] D. Boneh and M. Zhandry, "Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation," in *Advances in Cryptology—CRYPTO 2014—Proceedings of the 34th Annual Cryptology Conference*, pp. 480–499, Santa Barbara, CA, USA, August 2014.

[4] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *Proceedings of the 40th Annual Symposium on the Foundations of Computer Science*, pp. 120–130, IEEE, New York, NY, USA, October 1999.

[5] M. Liskov, "Updatable zero-knowledge databases," in *Advances in Cryptology—ASIACRYPT 2005, Proceedings of the 11th International Conference on the Theory and Application of Cryptology and Information Security*, pp. 174–198, Chennai, India, December 2005.

[6] M. Belenkiy, M. Chase, M. Kohlweiss, and A. Lysyanskaya, "Compact e-cash and simulatable VRFs revisited," in *Pairing-Based Cryptography—Pairing 2009, Proceedings of the Third International Conference*, pp. 114–131, Palo Alto, CA, USA, August 2009.

[7] G. Fuchsbauer, "Constrained verifiable random functions," in *Security and Cryptography for Networks—Proceedings of the 9th International Conference, SCN 2014*, pp. 95–114, Amalfi, Italy, September 2014.

[8] S. Micali and R. L. Rivest, "Micropayments revisited," in *Topics in Cryptology—CT-RSA 2002, Proceedings of the the Cryptographer's Track at the RSA Conference*, pp. 149–163, San Jose, CA, USA, February 2002.

[9] S. Hohenberger, V. Koppula, and B. Waters, "Adaptively secure puncturable pseudorandom functions in the standard model," in *Advances in Cryptology—ASIACRYPT 2015—Proceedings of the 21st International Conference on the Theory and Application of Cryptology and Information Security*, pp. 79–102, Auckland, New Zealand, November-December 2015.

[10] B. Barak, O. Goldreich, R. Impagliazzo et al., "On the (Im) possibility of obfuscating programs," in *Advances in Cryptology—CRYPTO 2001*, pp. 1–18, Springer, Berlin, Germany, 2001.

[11] A. Lysyanskaya, "Unique signatures and verifiable random functions from the DH-DDH separation," in *Advances in Cryptology—CRYPTO 2002, Proceedings of the 22nd Annual*

International Cryptology Conference, pp. 597–612, Santa Barbara, CA, USA, August 2002.

[12] Y. Dodis and A. Yampolskiy, "A verifiable random function with short proofs and keys," in *Public Key Cryptography—PKC 2005, Proceedings of the 8th International Workshop on Theory and Practice in Public Key Cryptography*, pp. 416–431, Les Diablerets, Switzerland, January 2005.

[13] S. Hohenberger and B. Waters, "Constructing verifiable random functions with large input spaces," in *Advances in Cryptology—EUROCRYPT 2010, Proceedings of the 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 656–672, French Riviera, France, May-June 2010.

[14] M. Abdalla, D. Catalano, and D. Fiore, "Verifiable random functions: relations to identity-based key encapsulation and new constructions," *Journal of Cryptology*, vol. 27, no. 3, pp. 544–593, 2014.

[15] G. Fuchsbauer, M. Konstantinov, K. Pietrzak, and V. Rao, "Adaptive security of constrained PRFs," in *Advances in Cryptology—ASIACRYPT 2014*, Springer, Berlin, Germany, 2014.

[16] D. Hofheinz, A. Kamath, V. Koppula, and B. Waters, "Adaptively secure constrained pseudorandom functions," in *Financial Cryptography and Data Security*, Springer, Cham, Switzerland, 2019.

[17] A. Kiayias, S. Papadopoulos, N. Triandopoulos, and T. Zacharias, "Delegatable pseudorandom functions and applications," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security*, pp. 669–684, CCS'13, Berlin, Germany, November 2013.

[18] E. Boyle, S. Goldwasser, and I. Ivan, "Functional signatures and pseudorandom functions," in *Public-Key Cryptography—PKC 2014—Proceedings of the 17th International Conference on Practice and Theory in Public-Key Cryptography*, pp. 501–519, Buenos Aires, Argentina, March 2014.

[19] N. Chandran, S. Raghuraman, and D. Vinayagamurthy, "Constrained pseudorandom functions: verifiable and delegatable," in *Security and Cryptography for Networks*, Springer, Cham, Switzerland, 2014.

[20] D. Boneh and X. Boyen, "Secure identity based encryption without random oracles," in *Advances in Cryptology—CRYPTO 2004, Proceedings of the 24th Annual International CryptologyConference*, pp. 443–459, Santa Barbara, CA, USA, August 2004.

[21] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters, "Candidate indistinguishability obfuscation and functional encryption for all circuits," in *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science*, pp. 40–49, FOCS, Berkeley, CA, USA, October, 2013.

[22] M. Chase, S. Meiklejohn, and Q. Déjà, "Using dual systems to revisit *q*-type assumptions," in *Advances in Cryptology—EUROCRYPT 2014—Proceedings of the 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 622–639, Copenhagen, Denmark, May 2014.