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

Copyright © 2019 Muhua Liu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

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 $F: \mathcal{X} \times \mathcal{X} \rightarrow \mathcal{Y}$, for any polynomially chosen key $sk \in \mathcal{X}$, a polynomial probabilistic time (PPT) adversary cannot distinguish the output $F(sk, x)$ 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 $sk \in \mathcal{X}$ which can be used to evaluate all points that belonged to the domain $\mathcal{X}$. Given the master key $sk \in \mathcal{X}$ 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 $S_1, \ldots, S_q \subset \mathcal{X}$ and several function values at points $x_1, \ldots, x_q \in \mathcal{X}$ which were chosen adaptively by the adversary, the adversary cannot distinguish a function value $F(sk, x)$ from a random value for all $x \neq x_i, \forall i \in \{1, \ldots, q\}$, and $x \in \cap_{j=1}^q S_j$. 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
setup algorithm. Given a constrained key, the adversary must commit to a challenge point to achieve selective security—a weaker notion where the adversary must query a challenge point and (in 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 Hohenberger et al. [9], we construct a VRF with constrained keys for any set 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 $e : G_1 \times G_2 \to G_T$. The constrained VRFs map an input from $[0, 1]^l$ to an element of $G_T$. The secret key is a tuple $sk = (v, w, (d_1, \ldots, d_n), h)$, where $v \in G_1$, $w \in G_2$, $v \leftarrow Z_N$, $(d_1, \ldots, d_n, h) \leftarrow \{0, 1\}^n$ is an admissible hash function. VRFs are defined as

$$F(sk, x) = e\left(\prod_{i=1}^n d_i(x_i)w^i\right),$$

associated with a proof

$$P(sk, x) = (\prod_{i=1}^n d_i(x_i)w^i),$$

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

In order to verify the correctness of evaluation, we define a public key as $pk = (w, w', i\Theta(\ell'))$, where $i\Theta(\ell')$ is an obfuscation of a circuit which takes a point $x$ as input and outputs an element $D(x)$ for $G_T$. The verifier only needs to check $e(P(sk, x), w) = D(x)$ and $e(P(sk, x), w') = F(sk, x)$. 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 $\ell$–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.

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.

2. Preliminaries

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

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

$$\Pr[\forall i \leq Q(\lambda), H_u(x_i) = 1 \land H_u(z) = 0] \geq 1/\theta(Q(\lambda)),$$

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

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

Definition 2 (indistinguishability obfuscation $(\mathcal{O})$). 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

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

(ii) Indistinguishability: for any (not necessarily uniform) PPT distinguisher $\mathcal{D}$, there exists a negligible function $\text{negl}$ such that the following holds: if $\Pr[\mathcal{D}(\sigma, i\mathcal{O}(\lambda, C_0)) = 1 : (\sigma, C_0) \leftarrow \text{Samp}(1^\lambda)]$

$$= \Pr[\mathcal{D}(\sigma, i\mathcal{O}(\lambda, C_1)) = 1 : (\sigma, C_1) \leftarrow \text{Samp}(1^\lambda)] \leq \text{negl}(\lambda). \quad (5)$$

2.1. Assumptions. Let $\mathcal{C}$ be a PPT group algorithm that takes a security parameter $1^\lambda$ 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 $\lambda$–bit primes. $G_1, G_2$, and $G_T$ are groups of order $N = pq$, $e : G_1 \times G_2 \rightarrow G_T$ 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 $(N, G_p, G_q, G_1, G_2, G_T, e) \leftarrow \mathcal{C}(1^\lambda)$ and $b \leftarrow \{0, 1\}$. Let $T \leftarrow G_1$ if $b = 0$, else $T \leftarrow G_p$. The advantage of algorithm $\mathcal{A}$ in solving the subgroup decision problem is defined as

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

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

Assumption 2 (weak Bilinear Diffie–Hellman Inversion). Let $b \leftarrow \mathcal{C}(1^\lambda), g_1 \leftarrow G_1, a \leftarrow Z_N^*$, and $g^a \leftarrow G_2, y \leftarrow Z_N^*$. Let $D = (N, G_p, G_q, G_1, G_2, G_T, e, g_1, g_1^a, \ldots, g_1^a, g_2, g_2)$. Let $T = e(g_1^a, g_2)$ if $b = 0$, else
regarding to a set of function, where \( K \), \( Y \) for all \( \text{function} \), \( \text{where } K \) and \( Y \) is the range. \( F \) is said to be constrained VRFs with regard to a set \( S \subset X \) if there exists a constrained key space \( X' \), a proof space \( P \), and four algorithms (Setup, Constrain, Prove, and Verify):

(i) Setup \((1^\lambda) \rightarrow (pk, sk)\) : it is a PPT algorithm that takes the security parameter \( \lambda \) as input and outputs a pair of keys \((pk, sk)\), a description of the key space \( X' \), and a constrained key space \( X' \).

(ii) Constrain \((sk, S) \rightarrow sk_S\) : this algorithm takes the secret key \( sk \) and a set \( S \subset X' \) as input and outputs a constrained key \( sk_S \in X' \).

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

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

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

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

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

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

(i) \( y_0 = y_1 \),

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

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

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 \( \text{Exp}_{VRF}^{BDHI}(1^\lambda, b) \) for \( \lambda \in \mathbb{N} \):

(i) The challenger first chooses \( b \rightarrow \{0, 1\} \) and then generates \((pk, sk)\) by running the algorithm \( \text{Setup}(1^\lambda) \) 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 \( S \subset X' \), if \( V \cap S \neq \emptyset \), return \( sk_q \leftarrow \text{Constrain}(sk, S) \) and set \( V = V \cup S \); else return \( \perp \).

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

(3) Challenge: on input \( x^* \in X' \), 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 \( Y' \) if \( b = 1 \).

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

A constrained VRF is pseudorandom if for all PPT adversary \( \mathcal{A} \), it holds that

\[
\Pr[\text{Exp}_{VRF}^{BDHI}(1^\lambda, b) = 1] = 1/2 \leq \text{negl}(\lambda).
\]

### 3.4. Semiaadaptive Security
We give a weak definition for pseudorandomness which is called semiaadaptive 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 semiaadaptive security, it must satisfy selective security. Conversely, it may not be 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 \( F : X \times X' \rightarrow Y \) consists of four algorithms (Setup, Puncture, Prove, and Verify). The input domain is \( X \rightarrow [0, 1]^\ell \), where \( \ell = \ell(\lambda) \). The key space \( X \) and range space \( Y \) are defined as a part of the setup algorithm.

(i) Setup \((1^\lambda) \rightarrow (pk, sk)\) : On input the security parameter \( 1^\lambda \), run \((N, G_0, G_p, G_1, G_2, G_T, e) \rightarrow (X, 1) \) such that \( e : G_1 \times G_2 \rightarrow G_T \) and \( G_p \) and \( G_q \).
are subgroups of $G_1$. Let $n, \theta$ be polynomials such that there exists a \( \theta \)-admissible hash function $h : \{0, 1\}^{(\ell(\lambda))} \rightarrow \{0, 1\}^{(\ell(\lambda))}$.

The key space is $\mathcal{K} = G_1 \times G_2 \times \mathbb{Z}_N \times (\mathbb{Z}_N^\times)^n$, the range is $\mathcal{Y} = G_1$, and the proof space is $\mathcal{P} = G_1$. The setup algorithm chooses $\nu \in G_1, w \in G_2, y \in \mathbb{Z}_N$, and $(d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}) \in \mathbb{Z}_N^\times$ uniformly at random and sets $sk = (\nu, w, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$. The public key contains an obfuscation of a circuit $\mathcal{C}_1$, where $\mathcal{C}_1$ is described in Figure 1. Note that $\mathcal{C}_1$ has $\nu, w, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}), h$ hardwired in it. Set $pk = (w, w^\nu, i\theta(\mathcal{C}_1))$, where $\mathcal{C}_1$ 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,

$$F(sk, x) = e\left(\prod_{i=0}^{n} d_{i,0}^{b_i}, w\right) = P(sk, x) = e\left(\prod_{i=0}^{n} d_{i,0}^{b_i}\right). \tag{9}$$

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

(iii) Prove $(sk', x) \rightarrow (y, \pi)$ or $(\perp, \perp) :$ The punctured key $sk'$ is a program that takes an $\ell$-bit input $x$. We define

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

(iv) Verify $(pk, x, y, \pi) \rightarrow \{0, 1\} :$ To verify $(x, y, \pi) \in \{0, 1\}^{(\ell(\lambda))} \times G_T \times G_1$ with regard to $pk = (w, w^\nu, i\theta(\mathcal{C}_1))$, compute $D(x) = \mathcal{C}'_1(x) = e\left(\prod_{i=0}^{n} d_{i,0}^{b_i}, w\right)$ and output 1 if the following equations are satisfied:

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

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

4.1. Properties

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

$$e(\pi, w) = e\left(\prod_{i=0}^{n} d_{i,0}^{b_i}, w\right) = D(x),$$

$$e(\pi, w^\nu) = e\left(\prod_{i=0}^{n} d_{i,0}^{b_i}, w^\nu\right) = y = F(sk, x). \tag{12}$$

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

4.1.2. Uniqueness. Consider a public key $pk = (w, w^\nu, i\theta(\mathcal{C}_1))$, where $w \in G_2, y \in \mathbb{Z}_N$, and $\mathcal{C}_1$ is described in Figure 1. Given a value $x \in \{0, 1\}^{(\ell(\lambda))}$ and two pair values $(y_0, \pi_0)$ and $(y_1, \pi_1) \in G_T \times G_1$ that satisfy

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 $i\theta$ 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 $\mathcal{A}$. Without loss of generality, we assume that the adversary $\mathcal{A}$ 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, $\mathcal{A}$ 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 $(N, G_1, G_2, G_T, e) \leftarrow \mathcal{G}(1^\lambda)$, chooses $v \in G_1, w \in G_2, y \in \mathbb{Z}_N, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}) \in \mathbb{Z}_N^\times$, and sets $sk = (v, w, y, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$ and $pk = (w, w^\nu, i\theta(\mathcal{C}_1))$. Then, the challenger flips a coin $a \leftarrow \{0, 1\}$.
Table 1: The differences between each adjacent games.

| Game  | Key generation | Challenge answer |
|-------|----------------|------------------|
| Game 1 | \( sk = (v, w, y, \{(d_{i,0}, d_{i,1})\}_{i=1}^\alpha) \) | If \( H_u(x) = 1 \), \( y_0 = e(v^{\prod_i d_{i,0}^{\ell_i}}, w^\ell) \) |
| Game 2 | \( sk = (v, w, y, \{(d_{i,0}, d_{i,1})\}_{i=1}^\alpha) \) | If \( H_u(x^*) = 1 \), abort, else \( y_0 = e(v^{\prod_i d_{i,0}^{\ell_i}}, w^\ell) \) |
| Game 3 | \( sk' = (v, w, y, \{(\ell_i, o_i, e_i)\}_{i=1}^\alpha) \) where \( e_i = o_i \cdot a \), if \( H_u(x^*) = b \) | If \( H_u(x^*) = 1 \), abort, else \( y_0 = e(v_1^{\prod_i d_{i,0}^{\ell_i}}, w^\ell) \) |
| Game 4 | \( sk' = (v, w, y, \{(\ell_i, o_i, e_i)\}_{i=1}^\alpha) \) where \( V = (v_1, v_2, \ldots, v_n) \) | If \( H_u(x^*) = 1 \), abort, else \( y_0 = e(v_1^{\prod_i d_{i,0}^{\ell_i}}, w^\ell) \) |
| Game 5 | \( sk' = (v, w, y, \{(\ell_i, o_i, e_i)\}_{i=1}^\alpha) \) where \( V = (v_1, v_2, \ldots, v_n) \) | If \( H_u(x^*) = 1 \), abort, else \( y_0 = e(v_1^{\prod_i d_{i,0}^{\ell_i}}, w^\ell) \) |

(2) The adversary \( \mathcal{A} \) makes an 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 \( (F(sk, x_i), P(sk, x_i)) = (e(v^{\prod_i d_{i,0}^{\ell_i}}, w^\ell), v^{\prod_i d_{i,0}^{\ell_i}}) \).

(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 \( sk_{x^*} \leftarrow i\theta(\mathcal{C}_2) \) and \( h(x^*) = b_1^*, \ldots, b_n^* \). sets \( y_0 = e(v^{\prod_i d_{i,0}^{\ell_i}}, w^\ell) \) and \( y_1 \leftarrow G_T \), and returns \((pk, sk_{x^*}, y_0)\) to the adversary \( \mathcal{A} \).

(5) The adversary \( \mathcal{A} \) outputs a bit \( a^* \) and wins if \( a^* = a \).

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 \( w \in \{0, 1\}^n \) by the algorithm AdmSample of admissible hash function and aborts if either there exists an evaluation query \( x \) such that \( H_u(x) = 0 \) or the challenge query \( x^* \) such that \( H_u(x^*) = 1 \).

1. The challenger runs \((N, G_p, G_q, G_1, G_2, G_T, e) \leftarrow \mathcal{G}(1^\lambda)\), chooses \( v \in G_1, w \in G_2, y \in Z_N, (d_{1,0}, d_{1,1}) \), \( \ldots, (d_{n,0}, d_{n,1}) \in Z_N^2 \), and sets \( sk = (v, w, y, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1})) \) and \( pk = (w, v, i\theta(\mathcal{C}_2)) \). Then, the challenger flips a coin \( \alpha \leftarrow \{0, 1\} \) and runs \( u \leftarrow \text{AdmSample}(1^\lambda, Q) \).

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 \( h(x_i) \neq u_i \) 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 \((F(sk, x_i), P(sk, x_i)) = (e(v^{\prod_i d_{i,0}^{\ell_i}}, w^\ell), v^{\prod_i d_{i,0}^{\ell_i}}) \).

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 \( sk_{x^*} \leftarrow i\theta(\mathcal{C}_2) \) and \( h(x^*) = b_1^*, \ldots, b_n^* \). sets \( y_0 = e(v^{\prod_i d_{i,0}^{\ell_i}}, w^\ell) \) and \( y_1 \leftarrow G_T \), and returns \((pk, sk_{x^*}, y_0)\) to the adversary \( \mathcal{A} \).

5. The adversary \( \mathcal{A} \) outputs a bit \( a^* \) and wins if \( a^* = a \).

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

Proof. The difference between Game 1 and Game 2 is that we add an abort condition in Game 2. From the \( \theta \)-admissibility of hash function \( h \), we can get that for all \( x_1, \ldots, x_Q, x^* \), \( \Pr_\mathcal{A} \leftarrow \text{AdmSample}(1^{\lambda}, Q) \left[ \forall i, H_u(x_i) = 1 \land H_u(x^*) = 0 \right] \geq 1/\theta(Q(\lambda)) \). The two experiments are equal if Game 2 does not abort. Therefore, if \( \mathcal{A} \) wins with advantage \( e \) in Game 1, then it wins with advantage at least \( e/\theta(Q(\lambda)) \) 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 \( sk' \) which is randomly chosen from the key space. The detailed description is given as follows:

1. The challenger runs \((N, G_p, G_q, G_1, G_2, G_T, e) \leftarrow \mathcal{G}(1^\lambda)\), chooses \( v \in G_1, w \in G_2, y \in Z_N, (d_{1,0}, d_{1,1}) \), \( \ldots, (d_{n,0}, d_{n,1}) \in Z_N^2 \), and sets \( sk = (v, w, y, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1})) \) and \( pk = (w, v, i\theta(\mathcal{C}_1)) \). Then, the challenger flips a coin \( \alpha \leftarrow \{0, 1\} \) and runs \( u \leftarrow \text{AdmSample}(1^\lambda, Q) \).

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 \( h(x_i) \neq u_i \) 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 \((F(sk, x_i), P(sk, x_i)) = (e(v^{\prod_i d_{i,0}^{\ell_i}}, w^\ell), v^{\prod_i d_{i,0}^{\ell_i}}) \).
Input: a value \( x \)
Constants: \( v, w, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}), h \), 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 \( u \in \{0, 1\}^n \)

(a) compute \( h(x) = b_1, \ldots, b_n \);
(b) if \( H_w(x) = 0 \), then output \( D(x) = e(v_1^{\pi\cdot b_0}, w) \);
(c) else, output \( D(x) = e(v^{\pi\cdot b_0}, w) \).

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

(4) The challenger checks if \( H_w(x^*) = 0 \). If not, the game aborts. Else, the challenger chooses \( v_1 \in G_1, (e_{1,0}, e_{1,1}), \ldots, (e_{n,0}, e_{n,1}) \in Z_N^2 \), sets \( sk'' = (v_1, w, \gamma, (e_{1,0}, e_{1,1}), \ldots, (e_{n,0}, e_{n,1})) \), computes \( pk'' = (w, w', i \phi(\epsilon'_3), \pi) \), and sends \( y_0 = e(v_1^{\pi\cdot b_0}, w') \) and \( y_1 \longrightarrow G_2 \). Then, it returns (\( pk'', sk'', y'_0 \)) to the adversary \( A \).

(5) The adversary \( A \) outputs a bit \( a' \) and wins if \( a' = a \).

Lemma 2. Assuming \( i \phi \) 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 \( H_w(x) = 0 \). The detailed description is given as follows:

(1) The challenger runs \( (N, G_1, G_2, G_3, G_4, G_7, e) \rightarrow \mathcal{B}(1^\lambda) \), chooses \( v \in G_1, w \in G_2, \gamma \in Z_N, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}) \in Z_N^2 \), and sets \( sk = (v, w, \gamma, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1})) \) and \( pk = (w, w', i \phi(\epsilon'_3)) \). Then, the challenger flips a coin \( a \leftarrow \{0, 1\} \) and runs \( u \rightarrow \text{AdmSample}(1^\lambda, Q) \).

(2) The adversary \( A \) makes a evaluation query \( x_j \in \{0, 1\}^{Q(\lambda)} \). The challenger checks if \( H_w(x_j) = 1 \) (recall that \( H_w(x) = 0 \) if \( h(x_j) \neq u_j \) for all \( j \in [n] \)). If not, the game aborts. Else, the challenger computes \( h(x_j) = b_1', \ldots, b_n' \) and outputs \( (F(sk, x_i), P(sk, x_i)) = (e(v^{\pi\cdot b_0}, w), v^{\pi\cdot b_0}) \).

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

(4) The challenger checks if \( H_w(x^*) = 0 \). If not, the game aborts. Else, the challenger chooses \( v_1 \in G_1, (e_{1,0}, e_{1,1}), \ldots, (e_{n,0}, e_{n,1}) \in Z_N^2 \), sets \( sk'' = (v_1, w, \gamma, (e_{1,0}, e_{1,1}), \ldots, (e_{n,0}, e_{n,1})) \), computes \( pk'' = (w, w', i \phi(\epsilon'_3), \pi) \), and sends \( y_0 = e(v_1^{\pi\cdot b_0}, w') \) and \( y_1 \longrightarrow G_2 \). Then, it returns (\( pk'', sk'', y'_0 \)) to the adversary \( A \).

(5) The adversary \( A \) outputs a bit \( a' \) and wins if \( a' = a \).

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 \( \{e_{1,0}, e_{1,1}, \ldots, (e_{n,0}, e_{n,1})\} \) are chosen. In Game 3, \( \{e_{1,0}, e_{1,1}, \ldots, (e_{n,0}, e_{n,1})\} \) are chosen randomly from \( Z_N^2 \), while in Game 4, the challenger first chooses \( e_{1,0} \leftarrow Z_N \) and \( e_{1,1} \leftarrow Z_N \), and sets \( e_{1,0} = e_{1,1} \cdot a \), if \( h(x^*) = b \), else \( e_{1,0} = e_{1,1} \). Since \( a \leftarrow Z_N \) which is invertible with overwhelming probability, \( e_{1,0} = e_{1,1} \cdot a \) is a uniform element in \( Z_N \). 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 \( \mathcal{C}_3 \) and \( \mathcal{C}_4 \) is changed. The two circuits contain some constants \( \{v_1'\}^{n-1} \). When \( H_w(x) = 0 \), the related function values are computed using...
the constants \( \{v_i^1\}_{i=1}^n \). The detailed description is given as follows:

(1) The challenger runs \( (N, G_p, G_T, G_1, G_2, G_T, e) \rightarrow \mathcal{B}(1^\lambda) \), chooses \( v \in G_1, w \in G_2 \), \( \gamma \in \mathbb{Z}_N \), and \( (a_{1,0}, d_{1,1}), \ldots, (d_{a_{0}, d_{n,0}}, e_{n,1}) \in \mathbb{Z}_N^2 \). Then, the challenger computes \( sk = (v, w, \gamma, (a_{1,0}, d_{1,1}), \ldots, (d_{a_{0}, d_{n,0}}, e_{n,1})) \) and \( pk = (w, w^\gamma, ito(\mathcal{B})) \). Then, the challenger flips a coin \( \alpha \leftarrow \{0, 1\} \) and runs \( u \rightarrow \text{AdmSample}(1^\lambda, Q) \).

(2) The adversary \( \mathcal{A} \) makes an evaluation query \( x_i \in \{0, 1\}^\lambda \). The challenger checks if \( H_n(x_i) = 1 \) (recall that \( H_n(x) = 0 \) if \( h(x) \neq u_i \) for all \( i \in [n] \)). If not, the game aborts. Else, the challenger computes \( h(x_i) = b_i^1, \ldots, b_n^i \) and outputs \( (F(sk, x_i), P(sk, x_i)) = (e(v^i, w^{(i)}, w^\gamma), v^{i, \mathcal{B}, (\mathcal{B})}) \).

(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 chooses \( \gamma \in \mathbb{Z}_N \). If not, the game aborts. Else, the challenger chooses \( v_i \in G_1, a_i \in G_2, (e_i^1, e_i^2), \ldots, (e_i^{a_0}, e_i^{2n}) \in \mathbb{Z}_N \). Let \( V = (v_1, v_2, \ldots, v^\lambda_1) \). Set \( sk^i = (v_i, w, y_i, (e_i^1, e_i^2), \ldots, (e_i^{a_0}, e_i^{2n})) \), compute \( pk^i = (w, w^\gamma, ito(\mathcal{B})) \), \( h(x^*) = b_i^1, \ldots, b_n^i \), and \( D_{x^*} = e(v_i, a_i^{\mathcal{B}, (\mathcal{B})}, w) \). Set \( y_0 = e(v_i, a_i^{\mathcal{B}, (\mathcal{B})}, w) \) and \( y_1 \leftarrow G_T \), where the descriptions of \( \mathcal{C}_5 \) and \( \mathcal{C}_6 \) are given in Figures 5 and 6, respectively. Then, it returns \( (pk^i, sk^i, y_0) \) to the adversary \( \mathcal{A} \).

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

\textbf{Lemma 4.} Assuming \( ito \) is a secure indistinguishability obfuscator, Game 4 and Game 5 are computationally indistinguishable.

\textbf{Proof.} We will introduce an intermediate experiment \( \mathcal{A}_4 \) and prove that Game 4 and Game 4 are computationally indistinguishable and Game 4 and Game 5 are computationally indistinguishable.

The experiment \( \mathcal{A}_4 \) is the same as Game 5 except that \( sk \) 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, we construct a PPT adversary \( \mathcal{B} \) that breaks the \( ito \) 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: \( v_i \in G_1, a_i \rightarrow \mathbb{Z}_N, (e_i^1, e_i^2), \ldots, (e_i^{a_0}, e_i^{2n}) \in \mathbb{Z}_N \). Then \( sk = (v_i, w_i, y_i, (e_i^1, e_i^2), \ldots, (e_i^{a_0}, e_i^{2n})) \), \( V = (v_1, v_2, \ldots, v^\lambda_1) \). \( \mathcal{A} \) constructs circuits \( \mathcal{C}_0 = \mathcal{C}_5 \) and \( \mathcal{C}_1 = \mathcal{C}_6 \), where \( sk \) is replaced by \( sk' \). Then, \( \mathcal{A} \) sends \( \mathcal{C}_0 \) and \( \mathcal{C}_1 \) to the \( ito \) challenger and gets \( pk' = ito(\mathcal{B}) \). \( \mathcal{B} \) computes \( sk' = ito(\mathcal{B}), h(x^*) = b_i^1, \ldots, b_n^i, y_0 = e(v_i, a_i^{\mathcal{B}, (\mathcal{B})}) \), and \( y_1 \leftarrow G_T \) and returns \( (pk', sk', y_0) \) to the adversary \( \mathcal{A} \). \( \mathcal{A} \) outputs \( \alpha' \), if \( \alpha = \alpha' \) and \( \mathcal{B} \) outputs 0, else outputs 1.
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 2A. 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 $\alpha \leftarrow \{0,1\}$ and runs $u \leftarrow \text{AdmSample}(1^\lambda, Q)$. Then, the challenger runs $(N,G_p,G_q,G_1,G_2,G_T,\alpha) \leftarrow \mathbb{G}(1^\lambda)$, chooses $v \in G_1$, $w \in G_2$, $a, y \in \mathbb{Z}_7$, and $(d_{i,0},d_{i,1},\ldots,d_{n,0},d_{n,1}) \in \mathbb{Z}_7^n$, sets $\lambda = d_{i,b}$, $d_{i,b}$, and $sk = (v, w, y, (d_{i,0}^1, d_{i,1}^1), \ldots, (d_{n,0}^1, d_{n,1}^1))$ and $pk = (\lambda, a \cdot d_{i,b})$ and $\lambda \leftarrow \text{AdmSample}(1^\lambda, Q)$. The challenger checks if $H_u(x) = 1$ (recall that $H_u(x) = 0$, if $h(x) \neq u_j$ for all $j \in [n]$). If not, the game aborts. Else, the challenger computes $h(x) = b_1, \ldots, b_n$ and outputs $(P(sk, x_i), P(sk, x_i)) = (v, w, y, (d_{i,0}^1, d_{i,1}^1), \ldots, (d_{n,0}^1, d_{n,1}^1))$.

2. The adversary $A$ makes an evaluation query $x_i \in \{0,1\}$. The challenger computes $h(x) \leftarrow B_1, \ldots, B_n$ and outputs $(F(sk, x_i), F(sk, x_i)) = (e(v, w, y, (d_{i,0}^1, d_{i,1}^1), \ldots, (d_{n,0}^1, d_{n,1}^1)))$.

3. The adversary $A$ sends a challenge point $x^* \neq x_i$ for all $i \in [Q \lambda]$. The adversary $A$ outputs a bit $a'$ and wins if $a' = a$.

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

Proof. We observe the difference between Game 2 and Game 2A is the manner in which $d_{i,b}$ is chosen. In Game 2, $d_{i,b}$ is chosen randomly from $\mathbb{Z}_N$, while in Game 2A, the challenger first chooses $d_{i,b} \leftarrow \mathbb{Z}_N$ and $a \leftarrow \mathbb{Z}_N$ and sets $d_{i,b} = d_{i,b} \cdot a$. 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. $\square$

4.3.2. Game. 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 $v \leftarrow v \cdot a$. 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 $(v, v^a, \ldots, v^{a^{Q-1}})$.

1. The challenger flips a coin $\alpha \leftarrow \{0,1\}$ and runs $u \leftarrow \text{AdmSample}(1^\lambda, Q)$. Let $m(x) = [i : u_i \neq h(x_i)]$. Then, the challenger runs $(N,G_p,G_q,G_1,G_2,G_T,\alpha) \leftarrow \mathbb{G}(1^\lambda)$, chooses $v \in G_1$, $w \in G_2$, $a, y \in \mathbb{Z}_7$, and $(d_{i,0},d_{i,1},\ldots,d_{n,0},d_{n,1}) \in \mathbb{Z}_7^n$, sets $d_{i,b} = d_{i,b} \cdot a$. 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 related function values can only be computed by $(v, v^a, \ldots, v^{a^{Q-1}})$.

2. The adversary $A$ makes an evaluation query $x_i \in \{0,1\}$. The challenger checks if $H_u(x) = 1$ (recall that $H_u(x) = 0$, if $h(x) \neq u_j$ for all $j \in [n]$). If not, the game aborts. Else, the challenger computes $h(x) \leftarrow B_1, \ldots, B_n$, and outputs $y = e(v, w, y, (d_{i,0}^1, d_{i,1}^1), \ldots, (d_{n,0}^1, d_{n,1}^1))$, and $pk = (\lambda, a \cdot d_{i,b})$ and $\lambda \leftarrow \text{AdmSample}(1^\lambda, Q)$. Where the description of $\mathbb{G}_2$ is given in Figure 7.
Input: a value $x$
Constants: the element $w \in G_2, v' \in G_1, D = ((d_1^{(i,0)}, d_1^{(i,1)}), \ldots, (d_n^{(i,0)}, d_n^{(i,1)}))$, the random string $u \in \{0, 1, \pm\}^n$, and $V = (v, v', \ldots, v^{(i)})$

(a) compute $h(x_i) = b_1^{(i)}, \ldots, b_n^{(i)}$ and outputs $(F(sk, x_i), P(sk, x_i)) = (e(v^{(i)}, d_i^{(i,0)}, w^0), v^{(i)}, d_i^{(i,1)})).$

(b) if $H_u(x) = 0$, output $D(x) = e((v')^{\sum_j 1}, d_{j,0}, w)$;
(c) if $H_u(x) = 1$, output $D(x) = e((v')^{\sum_j 1}, d_{j,1}, w)$.

Lemma 7. Assuming $i\theta$ 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 $y_0 = e((v')^{\sum_j 1}, d_{j,0}, w^0)$ in Step 4.

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

Proof. We observe that the difference between Game $2_B$ and Game $2_C$ is that the term $v'^* \neq x_i$ 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 $v'$ 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 $2_C'$ and show that Game $2_C'$ and $2_C$ are computationally indistinguishable. Similarly, Game $2_C'$ and Game $2_D$ are computationally indistinguishable.

Game $2_C'$ is the same as Game $2_C$ except that $v$ is chosen from $G_p$. Suppose that there exists an adversary $\mathcal{A}$ which can distinguish Game $2_C'$ and $2_C$, we construct an adversary $\mathcal{B}$ that breaks assumption 1. $\mathcal{B}$ receives $(N, G_p, G_q, G_t, G_T, e, T)$, where $T \leftarrow G_1$ or $T \leftarrow G_p$. $\mathcal{B}$ sets $v = T$, chooses $v' \leftarrow G_1$, $w \leftarrow G_2, a, y \in \mathbb{Z}_N$, and $(d_1^{(i,0)}, d_1^{(i,1)}), \ldots, (d_n^{(i,0)}, d_n^{(i,1)}) \in \mathbb{Z}_N$ and computes $V, D$, and $pk$ as in Game $2_C$. Then, $\mathcal{B}$ runs the rest steps as in Game $2_C$. At last, $\mathcal{B}$ outputs $\alpha'$, if $\alpha = \alpha'$ and $\mathcal{B}$ guesses $T \in G_1$, else $\mathcal{B}$ guesses $T \in G_p$. Note that $\mathcal{B}$ simulates exactly Game $2_C'$ when $T \in G_1$, and $\mathcal{B}$ simulates exactly Game $2_C'$ when $T \in G_p$. Therefore, if there exists an adversary $\mathcal{B}$ that distinguishes $2_C$ and $2_C'$, with advantage $\epsilon$, there exists an adversary $\mathcal{B}$ 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 $sk'$. Otherwise, the related function values are computed by $sk$.

(1) The challenger flips a coin $\alpha \leftarrow \{0, 1\}$ and runs $u \leftarrow \text{AdmSample}(1^\lambda, Q)$. Let $m(x) = [i : u_i \neq h(x_i)]$. Then, the challenger runs $(N, G_p, G_q, G_t, G_T, e, T)$ $\leftarrow i\theta(1^\lambda)$, chooses $v \in G_p, v' \in G_q, w \in G_2, a, y \in \mathbb{Z}_N$, and $(d_1^{(0)}, d_1^{(1)}), \ldots, (d_n^{(0)}, d_n^{(1)}) \in \mathbb{Z}_N$, and sets $sk = (v, w, (d_1^{(0)}, d_1^{(1)}), \ldots, (d_n^{(0)}, d_n^{(1)}))$, $sk' = (v, w', (d_1^{(0)}, d_1^{(1)}), \ldots, (d_n^{(0)}, d_n^{(1)}))$, and $pk = (w, w', i\theta(C_3))$.

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

(3) The adversary $\mathcal{A}$ sends a challenge point $x^* \neq x_i$ for all $i \in \{0, 1\}^{\ell(x_i)}$.

(4) The challenger checks if $H_u(x^*) = 0$. If not, the game aborts. Else, the challenger computes $sk_{x^*} \leftarrow i\theta(C_3)$ and $h(x^*) = b_1^{(i)}, \ldots, b_n^{(i)}$, sets $y_0 = e((v')^{\sum_j 1}, d_{j,0}, w^0)$ and $y_1 \leftarrow G_T$, and returns $(pk, sk_{x^*}, y_0)$ to the adversary $\mathcal{A}$.

(5) The adversary $\mathcal{A}$ outputs a bit $\alpha'$ and wins if $\alpha' = \alpha$. □
Proof. We introduce two intermediate experiments $2_D$ and $2_{D_1}$ and show that Game 2$_D$ and 2$_{D_1}$ are computationally indistinguishable, Game 2$_{D_1}$ and Game 2$_D$ 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 $v, v', w, a, y$, and $d_{i,b}$ as in Game 2$_D$, sets $d_{i,b} = d'_{i,b}$ if $u_i = b$, else $d_{i,b} = a - d'_{i,b}$. Then, it answers the evaluation queries of $\mathcal{A}$ exactly as in Game 2$_D$. For the challenge queries, it sets $v' = (v')^{1/a}$ and computes $y_0 = e(v')^{\prod_{j} d_{j,b_0}}$, $w^0$ and $y_1 \rightarrow G_T$. The public key $pk$ is computed by the circuit $\mathcal{C}_4$, where $sk = (v, w^0, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$ and $sk' = (v', w^0, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$. The constrained key $sk_{x'}$ is computed as in Game 2$_{D_2}$.

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 $\mathcal{C}_4$.

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

Proof. We construct a PPT adversary $i\mathcal{O}$ that uses $2_{D_1}$ to break the security of $2_{B_1}$. $2_{B_1}$ runs Step 1 and Step 3 as in Game $v'$. On receiving the challenge point $G_1$, $y_0 = e((v')^{\prod_j d_{j,b_0}}, w^0)$ sets $\mathcal{A}$ and $2_{B_1}$ as in $2_{C_2}$ and constructs circuits $\mathcal{B}$ and $2_{C_2}$. Then, he sends $2_{C_2}$ to the $x''$ challenger and receives $G_1, 2_{B_1}$ computes $G_3, 2_{C_2}$ as in Game $v'$ and sends $G_3$ to $2_{C_2}$. $2_{C_2}$ returns $2_{C_1}$, if $2_{C_2}, 2_{C_1}$ outputs 0, else outputs 1.

Next, we only show that the circuit $2_{C_2}, 2_{D_2}$ have the identical functionality. For any $2_{C_2}$, such that $2_{C_2}$ For any $G_1$ such that $\mathcal{A}$. 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 $2_{C_2}$ that breaks the $2_{C_2}$ security.

Claim 2. Assuming $\mathcal{B}$ is a secure indistinguishability obfuscator, Game $\mathcal{B}$ 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 $T \leftrightarrow G_1$ and Game $T \leftrightarrow G_P$ are statistically indistinguishable.

Proof. The difference between Game $\mathcal{B}$ and Game $v = T$ is the chosen way of $v' \leftrightarrow G_1$, $w \leftrightarrow G_2, a, y \in Z_N$. In addition, $(d_{1,0, d_{1,1}}, \ldots, (d_{n,0}, d_{n,1})) \in Z_N^2$ is replaced by the value $V \cdot D$, and $pk$ in Game 2$_C$. Since $\mathcal{B}$, $a$ is invertible with overwhelming probability. Therefore, 2$_E$ is a uniform element from $\mathcal{A}$ and $\alpha'$ is also a uniformly random element from $\alpha = \alpha'$ in Game $\mathcal{B}$. It follows that the two experiments are statistically indistinguishable.

Lemma 11. Game 2$_{E}$ and Game 2$_{E'}$ are statistically indistinguishable.

Proof. The only difference between Game 2$_E$ and Game 2$_{E'}$ is the chosen way of the secret key $sk$ and $sk'$. In Game 2$_E$, the challenger chooses $v \in G_P, v' \in G_P, w \in G_2, y, d_{i,b} \in Z_N$ and sets $sk = (v, w^0, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$ and $sk' = (v', w^0, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$. While the challenger chooses $d_{i,b}, e_{i,b} \in Z_N$ and sets $sk = (v, w^0, (d_{1,0}, d_{1,1}), \ldots, (d_{n,0}, d_{n,1}))$ and $sk' = (v', w^0, (e_{1,0}, e_{1,1}), \ldots, (e_{n,0}, e_{n,1}))$ in Game 2$_{E'}$. Using Chinese remainder theorem, the
distributions \( \{\text{ } \bmod p, \text{ } \bmod q : x \leftarrow \mathbb{Z}_N \} \) and \( \{\text{ } \bmod p, \text{ } \bmod q : x, y \leftarrow \mathbb{Z}_N \} \) are statistically indistinguishable. Therefore, Game \( 2_E \) and Game \( 2_F \) are statistically indistinguishable. \( \square \)

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

Proof. The proof method is similar to Lemma 9. \( \square \)

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 \( \text{ for the polynomial set } S_i \), the challenger can guess the challenge point \( x^* \) with a probability of \( 1/|S_i| \). 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 \( F : \mathcal{X} \times \mathcal{X} \rightarrow \mathcal{Y} \) be a puncturable VRF (Setup, Puncture, Prove, and Verify), and \( P : \mathcal{X} \times \mathcal{X} \rightarrow \mathcal{G}_1 \) be a function of generation proof. We construct constrained VRFs \( (F, \text{Setup}, F, \text{Constrain}, F, \text{Prove}, F, \text{Verify}) \) by invoking the puncturable VRFs:

(i) \( F.\text{Setup}(1^k) \rightarrow (pk, sk) : \text{Run the algorithm } (pk_1, sk_1) \leftarrow \text{Setup}(1^k). \text{ Set } pk = pk_1 \text{ and } sk = sk_1. \)

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

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

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

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 \( (\text{Setup}, \text{Puncture}, \text{Prove}, \text{and Verify}) \) 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. \( \square \)

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 \( b \leftarrow \{0,1\} \) and then generates \( (pk, sk) \) by running the algorithm \( F.\text{Setup}(1^k) \)

(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 \( sk_{S_j} \leftarrow i\mathcal{O}(C_{sk,S_j}) \)

(iii) \( \mathcal{A} \) sends a challenge query \( x^* \) such that \( x^* \neq x_i \) for all \( i \leq q_1 \) and \( x^* \neq x_i \) for all \( j \leq q_2 \). Then, the challenger sets \( y_i = F(sk, x^*) \) and \( y_{i} \leftarrow \mathcal{Y} \) and outputs \( (y_{i}, pk) \)

(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 \( S_i \), the challenger guesses a challenge query \( x' \in S_i \). 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 \( b \leftarrow \{0,1\} \) and then generates \( (pk, sk) \) by running the algorithm \( F.\text{Setup}(1^k) \)

(ii) The adversary makes evaluation queries or constrained queries: For the first constrained query \( S_i \), the challenger chooses \( x' \leftarrow S_i \) and output \( sk_{S} \leftarrow i\mathcal{O}(C_{sk,s}) \). 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 $\mathcal{C}_{sk,S}$.](image)

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

(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 $j \leq q_2$. If $x^* \neq x_i'$, the experiment aborts. Else, the challenger sets $y_0 = F(sk, x^*)$ and $y_1 \leftarrow y'$, computes, and outputs $(y_0, pk)$.

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

**Lemma 13.** For any PPT adversary $\mathcal{A}$, if $\mathcal{A}$ wins with advantage $\varepsilon$ in Game 1, then it wins with advantage $\varepsilon/|S_i|$ 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 $1/|S_i|$, if $\mathcal{A}$ wins with advantage $\varepsilon$ in Game 1, then it wins with advantage $\varepsilon/|S_i|$ 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 $sk_{x_i}$ 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 $b \leftarrow \{0, 1\}$ and then generates $(pk, sk)$ by running the algorithm $F$.Setup ($1^{|}$).

(ii) The adversary makes evaluation queries or constrained queries: For the first constrained query $S_i$. The challenger chooses $x' \leftarrow S_i$, computes $sk_{x'} \leftarrow i\mathcal{O}(\mathcal{C}_{sk,S})$ and $\pi = \text{Puncture}(sk, x')$, and outputs $sk_{S_i} \leftarrow i\mathcal{O}(\mathcal{C}_{sk,S})$, where the description of the circuit $\mathcal{C}_{sk,S}$ 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 $\mathcal{A}$ sends an evaluation query $x_i$ such that $x_i = x_i'$, the experiment aborts. Else, if $x_i \neq x_i'$, output $(F(sk, x_i), P(sk, x_i)) = \text{Prove}(sk_{x_i'}, x_i)$

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

(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 $j \leq q_2$. If $x^* \neq x_i'$, the experiment aborts. Else, the challenger sets $y_0 = F(sk, x^*)$ and $y_1 \leftarrow y'$, computes, and outputs $(y_0, pk)$.

(iv) $\mathcal{A}$ 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 $i$th constrained query. In Game 2$_{i-1}$, $sk_{S_j} \leftarrow i\mathcal{O}(\mathcal{C}_{sk,S})$, while in Game 2$_i$, $sk_{S_j} \leftarrow i\mathcal{O}(\mathcal{C}_{sk,S})$. In order to prove that the two games are indistinguishable, we only need to show that the circuit $\mathcal{C}_{sk,S}$ and $\mathcal{C}_{sk,S}$ are functionally identical.

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

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

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 $\mathcal{Y}'$.

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

(ii) The adversary makes evaluation queries or constrained queries: For the first constrained query $S_i$, the challenger chooses $x' \leftarrow S_i$ and output $sk_{S_i} \leftarrow i\mathcal{O}(\mathcal{C}_{sk,S})$. 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 $x_i = x_i'$, the experiment aborts. Else, if $x_i \neq x_i'$, output $(F(sk, x_i), P(sk, x_i))$

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

(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 $j \leq q_2$. If $x^* \neq x_i'$, the experiment aborts. Else, the challenger sets $y_0 \leftarrow y'$ and $y_1 \leftarrow y'$ and outputs $(y_0, pk)$.

(iv) $\mathcal{A}$ outputs $b'$ and wins if $b = b'$.
Lemma 15. **Assuming the puncturable VRFs are secure, Game 2_{\Pi} and Game 3 are computationally indistinguishable.**

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

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

We observe that both $y_0$ and $y_1$ are chosen randomly from $\mathcal{Y}$. Therefore, for any PPT adversary $A$, 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 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 Cryptology Conference, 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.
