Candidate Differing-Inputs Obfuscation from Indistinguishability Obfuscation and Auxiliary-Input Point Obfuscation

Pan Dongxue[1,2], Li Hongda[1,2], Ni Peifang[1,2]

[1] The Data Assurance and Communication Security Research Center,
Chinese Academy of Sciences, Beijing 100093, China
[2] State Key Lab of Information Security, Institute of Information Engineering,
School of Cyber Security, Chinese Academy of Sciences, Beijing 100093, China
pandongxue@iie.ac.cn, lihongda@iie.ac.cn, nipeifang@iie.ac.cn

Abstract. Differing-inputs obfuscation (diO), first proposed by Barak et. al. [4], provides stronger security than that provided by indistinguishability obfuscation (iO). An iO scheme provides indistinguishability between the obfuscations of two programs that are equivalent and have the same length of description. A diO scheme ensures that the obfuscations of two efficiently generated programs with the same description length are indistinguishable if it is hard to find an input on which their outputs differ. Ananth et. al. [1] showed the definition of diO with respect to arbitrary auxiliary inputs. However, Garg et al. [19] showed that the existence of this kind of diO contradicts a certain "special-purpose obfuscation" conjecture. Ishai, Pandey and Sahai [23] suggested a diO variant called public-coin diO, which requires the auxiliary input to be a public random string and given as input to all relevant algorithms. They gave a construction of public-coin diO by assuming the existence of public-coin differing-inputs obfuscator for NC^1 circuits.
In this paper, we use a slightly different definition, called public-coin-dependent diO. It allows the obfuscation algorithm to additionally take as input the random coins used to sample the circuit pair (including the circuit to be obfuscated) and thus the obfuscation algorithm can use the property of the circuit pair. We first construct a public-coin differing-inputs obfuscator for a class of new defined function with iO and point obfuscation with auxiliary input (AIPO). And then we use it to complete the public-coin-dependent diO for any pair of circuits that are hard to be found an input on which their outputs differ. The constructions are based on secure iO schemes for NC^1, fully homomorphic encryption scheme, and the existence of AIPO. Besides, we show the applications of our constructions.

Keywords: Differing-inputs obfuscation, indistinguishability obfuscation, point function, auxiliary input.

1 Introduction

Program obfuscation can make a program "unintelligible" while preserving its functionality. Barak et al. [4] first proposed the notion of virtual black-box (VBB) obfuscation. VBB obfuscation requires that the obfuscation of one arbitrary function leaks nothing except what can be learnt from a black-box oracle access to the function. However, Barak et al. [4] also showed a family of circuits called inherently unobfuscatable function that cannot be VBB obfuscated. In light of this impossibility result, Barak et al. presented weaker notions of obfuscation such as indistinguishability obfuscation and differing-inputs obfuscation and left open the problem of realizing such weaker notions.

Indistinguishability obfuscation. Indistinguishability obfuscation (iO) [4] requires that for any two equivalent circuits C_0 and C_1 of similar poly-size, any probabilistic polynomial-time (PPT) adversarial algorithm can distinguish between the obfuscations of C_0 and C_1 with negligible probability. Garg et al. [18] showed a candidate construction of indistinguishability obfuscator for all poly-size circuits and applied it to the construction of functional encryption for all circuits. They first transformed an NC^1 circuit into matrix branching programs and used a simplified variant of multilinear maps, called Multilinear Jigsaw Puzzles, to describe a candidate construction for indistinguishability obfuscation for NC^1 circuits. And then they used it and fully homomorphic encryption (FHE) scheme with with decryption in NC^1 to construct the construction of indistinguishability obfuscator for all poly-size circuits. After this work, [2, 11] presented new constructions of indistinguishability obfuscation for NC^1 circuits. However, [29, 25] analyzed the multilinear map used in [18] and found attacks on it from zero-encoding. Fortunately, [30] proposed a new way to obfuscate NC^1 circuits, via composite-order multilinear maps. The construction operates directly on straight-line NC^1 circuits, rather than converting them to matrix branching programs as in other known approaches.

Also based on composite-order graded encoding schemes, [10] presented a candidate obfuscator that operates directly on poly-size circuits. And based on a new "weak multilinear map model" [17] that captures all known polynomial-time attacks on [18], [22] gave a new iO candidate which can be seen as a small modification or generalization of the original candidate of [18]. These positive results on iO support our construction of public-coin-dependent diO.

Differing-inputs obfuscation. The stronger notion of differing-inputs obfuscation [4] states that if there exists a PPT adversary that can distinguish between obfuscations of circuits C_0 and C_1 with non-negligible probability, then there exists a PPT adversary that can actually extract an input on which the two circuits differ. Bellare, Stepanovs, and Tessaro [5] showed a definitional framework for diO. Instead of applying for all circuits, the security of obfuscation in the framework is parameterized by a class of samplers. Based on different types of samplers, one can define and study corresponding restricted forms of diO. Ananth et. al. [1] showed the definition of diO with respect to arbitrary auxiliary inputs. However, Garg et al. [19] showed that the existence of this kind of diO contradicts a certain "special-purpose obfuscation" conjecture.

Ishai, Pandey and Sahai [23] suggested a diO variant called public-coin diO. A public-coin diO requires the auxiliary input to be a public random string which is given as input to all relevant algorithms. They gave a construction of public-coin diO for polynomial-time Turing Machines (TMs) by assuming the existence of public-coin differing-inputs obfuscator for NC^1 circuits, public-coin succinct non-interactive arguments of knowledge, and fully homomorphic encryption scheme with decryption in NC^1. Based on the existence of public-coin collision-resistant hash functions, Boyle and Pass [9] showed that public-coin diO and extractable one-way functions (EOWF) with some carefully defined distributional auxiliary input cannot exist simultaneously. However, to the best of our knowledge, there is no work presenting constructions of EOWFs with arbitrary auxiliary input or presenting constructions that deny the existence of public-coin diO.

The recent work in [6] provides negative results on sub-exponential security and polynomial security for diO which are both based on sub-exponentially secure assumptions. One is that sub-exponentially secure diO for TMs does not exist if sub-exponentially secure one-way functions exist, and the other is that polynomially secure diO for TMs does not exist if in addition sub-exponentially secure iO exists. Unless otherwise stated, the obfuscation schemes (diO, iO etc.) in this paper is polynomially secure.

To the best of our knowledge, the previous works on constructing diO algorithm ignore the availability of the property of the circuit pair in the construction. In this paper, we use a slightly different definition of public-coin diO, called public-coin-dependent diO. It has the same security as public-coin diO, but it allows the obfuscation algorithm to additionally take as input the random coins used to sample the circuit pair (including the circuit to be obfuscated) and thus the obfuscation algorithm can use the property of the circuit pair. In this paper, we present a construction of polynomially secure public-coin-dependent diO for polynomial-time circuits based on fully homomorphic encryption scheme (FHE), polynomially secure iO schemes for NC^1 circuits, and point obfuscation with auxiliary input (AIPO). The first two tools implies the polynomially secure iO [18], which is a main tool used in our construction.

Point obfuscation with auxiliary input. Point obfuscation [13, 24, 28] is an obfuscation scheme of the class of point function I_x that outputs 1 on input x and outputs 0 otherwise. Point obfuscation requires that an PPT adversary with the given obfuscation can learn nothing more than what can be learnt with a black-box oracle access to the point function. [21] proves the impossibility of obfuscation with auxiliary input, however, there is no negative result on the existence of AIPO. Worst-case AIPO [8] requires that even the adversary is given some "leakage" about x, so-called auxiliary information, the obfuscation with auxiliary input does not leak anything more than what can be learnt with the auxiliary information and a black-box oracle access to the point function. A weaker distributional definition for AIPO is auxiliary input point obfuscation for unpredictable distributions (see Definition 2.3). The Proposition A.2 in [7] states that any VBB obfuscator with auxiliary input is also c-self-composable for any constant c. We can obtain that AIPO is c-self-composable for any constant c by the same proof method.

There are works [14, 16] on obfuscating multi-bit point function $I_{x,y}$ that outputs y on input x and outputs 0 otherwise. Multi-bit point obfuscation with auxiliary input (MB-AIPO) is proved to contradict the existence of iO [12]. This is the reason why we choose to assume the existence of AIPO as opposed to MB-AIPO for the construction of diO.

1.1 Our Results

To the best of our knowledge, all the previous constructions of diO based on the existence of some diO algorithm, for instance, [23] constructs public-coin diO with assuming the existence of public-coin differing-inputs obfuscator for NC^1 circuits. And the previous works on constructing diO algorithm ignore the availability of the property of the

circuit pair. In this paper, we define public-coin-dependent diO with the same security as public-coin diO. It allows the obfuscation algorithm to additionally take as input the random coins used to sample the circuit pair (including the circuit to be obfuscated) and thus the obfuscation algorithm can use the property of the circuit pair. We first construct a public-coin diO algorithm for a specially defined circuit class, and then we use it to achieve our public-coin-dependent diO algorithm. Besides, we show the applications of our constructions.

In order to construct public-coin-dependent diO for any pair of efficiently generated programs with the same size that are hard to be found an input on which their outputs differ, we first construct public-coin diO for the class of hiding-input point function with multi-bit output $\{I_{\varphi,y}\}_{\varphi,y\in\{0,1\}^{n(\lambda)}}$, where $\varphi \in L$ ($L \in NP$) is a hard problem and $I_{\varphi,y}$ outputs y on input x such that $\mathcal{R}_L(\varphi, x) = 1$ and outputs 0 otherwise (See the meaning of the symbols in section 2 and section 3).

Let $AIPO$ be an auxiliary input point obfuscation for unpredictable distributions. To begin the construction of public-coin diO for a sampled $I_{\varphi,y}$, where the length of y is n and $y = y_1, y_2, \cdots, y_n$, we claim that $AIPO$ is c-self-composable for any constant c. Then we also use $AIPO$ to denote an obfuscation of constant-bit point function (similar to multi-bit point function). We first choose a constant c and divide y into $q = q(\lambda)$ parts such that $(q-1)c < n \leq qc$ and we have

$$\begin{cases} v_i = y_{(i-1)c+1}, \cdots, y_{ic},\ i \in [q-1] \\ v_q = y_{(q-1)c+1}, \cdots, y_n,\ i = q \end{cases}.$$

Then we randomly choose $x_1, \cdots, x_q \in \{0,1\}^\lambda$. And I_{x_i,v_i} outputs v_i on input x_i and outputs 0 otherwise, $I_{\varphi,x_1||\cdots||x_q}$ outputs $x' = x_1||\cdots||x_q$ on input x such that $\mathcal{R}_L(\varphi, x) = 1$ and outputs 0 otherwise. We compute $o_i = AIPO(I_{x_i,v_i})$ and $aux = iO(I_{\varphi,x_1||\cdots||x_q})$. Here $AIPO$ is an obfuscation scheme for I_{x_i,v_i} with auxiliary input. On input (M_L, x), the evaluation algorithm $Evaluate_L$ first computes $aux(x)$ and outputs 0 if $aux(x) = 0$, otherwise it divides $aux(x) = x_1, \cdots, x_q$ and outputs $y = o_1(x_1), \cdots, o_q(x_q)$. Hence then, we obtain a diO scheme for $I_{\varphi,y}$ that

$$diO_L(I_{\varphi,y}) = (M_L = (aux, o_1, \cdots, o_q), Evaluate_L).$$

(see details in section 3)

With this result, we complete the construction of public-coin-dependent diO for any efficiently generated circuit pair $(C_0, C_1) \leftarrow Samp(r)$, where $Samp$ is a public-coin differing-inputs sampler (see definition 2.5) and r is a random value. We define $L = (L_Y, L_N)$ as follows:

$$\begin{cases} L_Y = \{(C_0, C_1) : \exists\, x \ such\ that\ C_0(x) \neq C_1(x)\} \\ L_N = \{(C_0, C_1) : C_0(x) = C_1(x)\ for\ all\ x\} \end{cases},$$

where $(C_0, C_1) \leftarrow Samp(r)$, $r \in_R \{0,1\}^{poly(\lambda)}$. Hence then, L is an NP language. Then, for any circuit pair $(C_0, C_1) \in L$, it is hard to find an input x such that $C_0(x) \neq C_1(x)$, then $\varphi = (C_0, C_1) \in L$ is a hard problem. And then, we can use the public-coin diO for $I_{\varphi,y}$ and FHE schemes to construct our public-coin-dependent diO.

1.2 Outline

In section 2, we define the notations and definitions that are used through the paper. In section 3, we define a new function called hiding-input point function with multi-bit output, prove the constant-self-composability of AIPO, and construct a public-coin diO scheme for the class of hiding-input point functions. In section 4, we complete the construction of public-coin-dependent diO using the result obtained in section 3. In section 5, we show the applications of our constructions. In section 6, we make a conclusion of our work.

2 Preliminaries

2.1 Notations

Let $A(\cdot)$ be a probabilistic algorithm and let $A(x)$ be the result of running algorithm A on input x, then we use $y = A(x)$ (or $y \leftarrow A(x)$) to denote that y is set as $A(x)$. Let $A_r(x)$ be the result of running algorithm A on input x with random value r. For a finite set (distribution) $\mathcal{S}$, we use $y \in_R S$ (or $y \leftarrow_R S$) to denote that y is uniformly selected from $\mathcal{S}$. For any promise problem (language) $L = (L_Y, L_N)$, where L_Y is the collection of the yes instances

and L_N is the collection of the no instances, and for any instance $x \in L$, we denote by $\mathcal{R}_L$ the efficiently computable binary NP relation for L. And for any witnesses w of $x \in L_Y$, $\mathcal{R}_L(x, w) = 1$. We use $[l]$ to denote the set $\{1, 2, \cdots, l\}$. We write $negl(\cdot)$ to denote an unspecified negligible function, $poly(\cdot)$ an unspecified polynomial. We denote by $a||b$ the concatenation of two bit strings a and b. We use "$X \stackrel{c}{=} Y$" to denote that probabilistic distributions X and Y are computationally indistinguishable. Unless otherwise stated, we use λ to denote the security parameter.

2.2 Obfuscation

Definition 2.1 ***Virtual black-box (VBB) obfuscation [3].*** *A probabilistic algorithm O is a circuit obfuscator if the following three conditions hold:*

- *(functionality) For every circuit C, the string $O(C)$ describes a circuit that computes the same function as C.*
- *(polynomial slowdown) There is a polynomial p such that for every circuit C, $|O(C)| \leq p(|C|)$.*
- *(virtual black-box property) For any probabilistic polynomial-time (PPT) A, there is a PPT S and a negligible function $negl(\cdot)$ such that for all circuits C*

$$\left|\Pr[A(O(C)) = 1] - \Pr[S^C(1^{|C|}) = 1]\right| \leq negl(|C|).$$

We say that O is efficient if it runs in polynomial time.

Definition 2.2 ***Unpredictable distribution [12].*** *A distribution ensemble $\mathcal{D} = \{\mathcal{D}_\lambda = (\mathcal{Z}_\lambda, \mathcal{X}_\lambda)\}_{\lambda \in N}$ on pairs of strings is unpredictable if no poly-size circuit family can predict $\mathcal{X}_\lambda$ from $\mathcal{Z}_\lambda$. That is, for every poly-size circuit sequence $\{\mathcal{C}_\lambda\}_{\lambda \in N}$ and for all large enough λ:*

$$\Pr_{(z,x) \leftarrow_R D_\lambda}[\mathcal{C}_\lambda(z) = x] \leq negl(\lambda).$$

Definition 2.3 ***Auxiliary input point obfuscation for unpredictable distributions (AIPO) [12].*** *A PPT algorithm AIPO is a point obfuscator for unpredictable distributions if it satisfies the functionality and polynomial slowdown requirements as in VBB-obfuscation, and the following secrecy property: for any (efficiently samplable) unpredictable distribution $\mathcal{B}_1$ over $\{0,1\}^{poly(\lambda)} \times \{0,1\}^\lambda$, it holds for any PPT algorithm $\mathcal{B}_2$ that the probability of outputting true by the following experiment for $(\mathcal{B}_1, \mathcal{B}_2)$ is negligibly close to $1/2$:*

$b \leftarrow_R \{0,1\}$,
$(z, x_0) \leftarrow_R \mathcal{B}_1(1^\lambda)$,
$x_1 \leftarrow_R \{0,1\}^\lambda$,
$p \leftarrow_R AIPO(I_{x_b})$,
$b' \leftarrow_R \mathcal{B}_2(1^\lambda, p, z)$,
return $b = b'$.

The probability is over the coins of adversary $(\mathcal{B}_1, \mathcal{B}_2)$, the coins of AIPO and the choices of x_1 and b.

Definition 2.4 ***Indistinguishability obfuscation (iO) [18].*** *A PPT algorithm iO is called an indistinguishability obfuscator for a circuit ensemble $\{\mathcal{C}_\lambda\}_{\lambda \in N}$ if the following conditions are satisfied:*

- *(functionality) For all security parameters $\lambda \in N$, for all $C \in \mathcal{C}_\lambda$, and for all input x we have that*

$$\Pr[C'(x) = C(x) : C' \leftarrow iO(1^\lambda, C)] = 1.$$

- *(security) For any PPT distinguisher D, there exists a negligible function $negl(\cdot)$ such that the following holds: For all security parameters $\lambda \in N$, for all pairs of same size circuits $C_0, C_1 \in \mathcal{C}_\lambda$, we have that if $C_0(x) = C_1(x)$ for all inputs x, then*

$$\left|\Pr[D(1^\lambda, iO(1^\lambda, C_0)) = 1] - \Pr[D(1^\lambda, iO(1^\lambda, C_1)) = 1]\right| \leq negl(\lambda).$$

Definition 2.5 ***Public-coin differing-inputs sampler for circuits [23].*** *An efficient non-uniform sampling algorithm $Sam = \{Sam_\lambda\}$ is called a public-coin differing-inputs sampler for the parameterized collection of circuits $\mathcal{C} = \{\mathcal{C}_\lambda\}$ if the output of Sam_λ is distributed over $\mathcal{C}_\lambda \times \mathcal{C}_\lambda$ and for every efficient non-uniform algorithm $A = \{A_\lambda\}$ there exists a negligible function $negl(\cdot)$ such that for all $\lambda \in N$:*

$$\Pr_r[C_0(x) \neq C_1(x) : (C_0, C_1) \leftarrow Sam_\lambda(r), x \leftarrow A_\lambda(r)] \leq negl(\lambda).$$

Definition 2.6 ***Public-coin differing-inputs obfuscator for circuits [23].*** *An uniform PPT algorithm diO is a public-coin differing-inputs obfuscator for the parameterized collection of circuits $\mathcal{C} = \{\mathcal{C}_\lambda\}$ if the following requirements hold:*

- *(functionality) For all security parameters $\lambda \in N$, for all $C \in \mathcal{C}_\lambda$, and for all input x we have that*

$$\Pr[C'(x) = C(x) : C' \leftarrow diO(1^\lambda, C)] = 1.$$

- *(security) For every public-coin differing-inputs samplers $Sam = \{Sam_\lambda\}$, for the collection $\mathcal{C}$, every PPT distinguishing algorithm $\mathcal{T} = \{\mathcal{T}_\lambda\}$, there exists a negligible function $negl(\cdot)$ such that for all security parameters $\lambda \in N$:*

$$\left|\Pr\left[\begin{array}{c}\mathcal{T}_\lambda(r, C') = 1 : (C_0, C_1) \leftarrow Sam_\lambda(r) \\ C' \leftarrow diO(1^\lambda, C_0)\end{array}\right]\right.$$
$$\left.-\Pr\left[\begin{array}{c}\mathcal{T}_\lambda(r, C') = 1 : (C_0, C_1) \leftarrow Sam_\lambda(r) \\ C' \leftarrow diO(1^\lambda, C_1)\end{array}\right]\right| \leq negl(\lambda),$$

  where the probability is taken over r and the coins of diO and $\mathcal{T}_\lambda$.

In section 4, we use a slightly different definition of public-coin diO by letting the obfuscation algorithm take as input the random value used to sample the circuit pair, which includes the circuit to be obfuscated. Then, it allows the obfuscation algorithm to use the property of the circuit pair and achieve the same security as the public-coin diO algorithm.

Definition 2.7 ***Public-coin-dependent differing-inputs obfuscator for circuits.*** *An uniform PPT algorithm diO is a public-coin-dependent differing-inputs obfuscator for the parameterized collection of circuits $\mathcal{C} = \{\mathcal{C}_\lambda\}$ if the following requirements hold:*

- *(functionality) For all security parameters $\lambda \in N$, for every public-coin differing-inputs samplers $Sam = \{Sam_\lambda\}$, for the collection $\mathcal{C}$, and for all input x we have that*

$$\Pr\left[\begin{array}{c}C'(x) = C(x) : (C_0, C_1) \leftarrow Sam_\lambda(r) \\ C \in \{C_0, C_1\} \\ C' \leftarrow diO(1^\lambda, C, r)\end{array}\right] = 1.$$

- *(security) As in Definition 2.6.*

2.3 Fully Homomorphic Encryption

Definition 2.8 ***Fully homomorphic encryption with decryption in NC^1 [23].*** *A fully homomorphic encryption (FHE) scheme is a public-key encryption scheme (Gen, Enc, Dec) with an additional evaluation algorithm $Eval$. $HFE = (Gen, Enc, Dec, Eval)$ is defined as following.*
Key generation: *The algorithm Gen takes the security parameter 1^λ and outputs a key pair (pk, sk), where sk is the secret decryption key and pk is the corresponding public encryption key.*
Encryption: *The algorithm Enc takes the public key pk, a message m and outputs a ciphertext $ct = Enc_{pk}(m)$.*
Decryption: *The algorithm Dec takes the secret key sk and a ciphertext ct and outputs a message $m' = Dec_{sk}(ct)$. It is guaranteed by the correctness of public-key encryption scheme that $Dec_{sk}(Enc_{pk}(m)) = m$.*
Evaluation: *The algorithm $Eval$ takes the public key pk, ciphertexts $c_1, \cdots, c_l$ corresponding to the bits $b_1, \cdots, b_l$ (under public key pk), a circuit $f : \{0,1\}^l \to \{0,1\}^{l'}$, and outputs a ciphertext as $ct' = Eval_{pk}(f, (c_1, \cdots, c_l))$ such that except with negligible probability over the randomness of all algorithms, the decryption of ct' is $f(b_1, \cdots, b_l)$, where $l = l(\lambda)$ and $l' = l'(\lambda)$ are arbitrary polynomials.*

A FHE scheme has decryption in NC^1 if there exists a constant $c \in N$ such that for all $\lambda \in N$ the depth of the circuit corresponding to the decryption function $Dec(1^\lambda, sk, \cdot)$ is at most $c \log \lambda$.

3 Public-coin Differing-inputs Obfuscator for Multi-bit Hiding-input Point Function

The goal in this section is to construct public-coin differing-inputs obfuscator for the class of hiding-input point function with multi-bit output, now we give the formal definition of this function class.

Definition 3.1 ***Hiding-input point function with multi-bit output.*** *Let L be a language in NP. $\mathcal{R}_L$ is the corresponding relation. Let $\varphi \in L$ be a hard problem such that for any PPT algorithm A, $\Pr[\mathcal{R}_L(\varphi, w) = 1 : w \leftarrow A(\varphi)] < negl(\lambda)$. Then, we define hiding-input point function with multi-bit output as follows:*

$$I_{\varphi,y}(x) = \begin{cases} y, \ if \ \mathcal{R}_L(\varphi, x) = 1 \\ 0, \ otherwise \end{cases},$$

where $y \in \{0,1\}^n$ and n is a polynomial in λ.

For example, let $G : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ be a pseudorandom generator, and then define L as

$$\begin{cases} L_Y = \{m : m \in \{0,1\}^{2\lambda}, \exists\, r \in \{0,1\}^\lambda \ such \ that \ m = G(r)\} \\ L_N = \{m : m \in \{0,1\}^{2\lambda}, \nexists\, r \in \{0,1\}^\lambda \ such \ that \ m = G(r)\} \end{cases},$$

then a hard problem $\varphi \in L$ is a real random $m \in \{0,1\}^{2\lambda}$.

3.1 Constant-self-composability of AIPO

To begin the construction of public-coin diO for a sampled $I_{\varphi,y}$, we claim that AIPO is c-self-composable for any constant c.

Claim 3.1 *Any auxiliary input point obfuscation for unpredictable distributions is c-self-composable for any constant c.*

Proof. For simplicity, we start with the case $c = 2$. Let O be an point obfuscator with auxiliary input for unpredictable distributions, and let A be a binary poly-size adversary and p a polynomial. For any (efficiently samplable) unpredictable distribution $\mathcal{D}_1 = (\mathcal{Z}_1, \mathcal{X}_\lambda)$ over $\{0,1\}^{poly(\lambda)} \times \{0,1\}^\lambda$, and for any $(z, x) \in_R \mathcal{D}_1$ and every $x' \in_R \{0,1\}^\lambda$, we set

$$\mathcal{D}_2 = (\mathcal{Z}_2, \mathcal{X}_\lambda) = \{((O_s(I_x), z), x) : (z, x) \in \mathcal{D}_1, s \in_R \{0,1\}^{poly(\lambda)}\}.$$

Here $O_s(I_x)$ is the result of running algorithm O on input I_x with randomness s. Then, by the property of point obfuscation with auxiliary input, no poly-size circuit family can predict $\mathcal{X}_\lambda$ from $\mathcal{Z}_2$. Then $\mathcal{D}_2$ is a new unpredictable distribution and we have that:

$$\left| \Pr_{A,r,s}[A(O_r(I_x), O_s(I_x), z) = 1] - \Pr_{A,r',s}[A(O_{r'}(I_{x'}), O_s(I_x), z) = 1] \right| \leq 1/2p(\lambda)$$

where we treated the second obfuscation as auxiliary input.

Now we can consider another unpredictable distribution

$$\mathcal{D}_3 = (\mathcal{Z}_3, \mathcal{X}_\lambda) = \left\{ \begin{array}{l} ((O_{r'}(I_{x'}), z), x) : (z, x) \in \mathcal{D}_1, \\ \qquad\qquad\qquad\quad x' \in_R \{0,1\}^\lambda, \\ \qquad\qquad\qquad\quad r' \in_R \{0,1\}^{poly(\lambda)} \end{array} \right\}.$$

Hence then, and for any $(z, x) \in_R \mathcal{D}_1$ and $x', x'' \in_R \{0,1\}^\lambda$, we have that:

$$\left| \Pr_{A,r',s}[A(O_{r'}(I_{x'}), O_s(I_x), z) = 1] - \Pr_{A,r',s'}[A(O_{r'}(I_{x'}), O_{s'}(I_{x''}), z) = 1] \right| \leq 1/2p(\lambda)$$

where we treated the first obfuscation as auxiliary input and $((O_{r'}(I_{x'}), z), x) \in_R \mathcal{D}_3$. Therefore,

$$\left| \Pr_{A,r,s}[A(O_r(I_x), O_s(I_x), z) = 1] - \Pr_{A,r',s'}[A(O_{r'}(I_{x'}), O_{s'}(I_{x''}), z) = 1] \right| \leq 1/p(\lambda)$$

and thus AIPO is 2-self-composable.

The result follows for $c = 2$ □

3.2 Construction of Unpredictable Distribution

Given a multi-bit hiding-input point function $I_{\varphi,y}$, we choose a constant c and divide $y = y_1, y_2, \cdots, y_n$ into $q = q(\lambda)$ parts such that $(q-1)c \leq n \leq qc$. Now we present a sequence of unpredictable distributions $\mathcal{D}^i$ over $\{0,1\}^{poly(\lambda)} \times \{0,1\}^{\lambda}$. Let iO be an indistinguishability obfuscator for all poly-size circuits initiated by the construction in [18]. We claim that

$$\mathcal{D}^i = (\mathcal{Z}_\lambda, \mathcal{X}_\lambda) = \left\{ \begin{array}{c} ((iO_r(I_{\varphi,x_1||\cdots||x_q}), i), x_i) \\ satisfying: x_1, \cdots, x_q \in_R \{0,1\}^{\lambda}, \\ r \in_R \{0,1\}^{poly(\lambda)} \end{array} \right\}$$

is unpredictable.

Claim 3.2 *Assuming the existence of polynomially secure indistinguishability obfuscator for NC^1 circuits and the existence of FHE with decryption in NC^1, and let iO be an indistinguishability obfuscator for all poly-size circuits initiated by the construction in [18]. In addition we assume that when $\varphi \in L_Y$, any PPT adversary without an witness for $\varphi \in L_Y$ cannot obtain x_i from $(iO_r(I_{\varphi,x_1||\cdots||x_q}), i)$, then $\mathcal{D}^i = (\mathcal{Z}_\lambda, \mathcal{X}_\lambda)$ defined above is an unpredictable distribution.*

Proof. In order to prove that $\mathcal{D}^i$ is unpredictable, we only need to prove that when $\varphi \in L_N$, the probability of computing x_i from the auxiliary input $(iO_r(I_{\varphi,x_1||\cdots||x_q}), i)$ is negligible. The result follows from the security of the indistinguishability obfuscator constructed in [18] and the hardness of the problem φ. Since when $\varphi \in L_N$, $(I_{\varphi,x_1||\cdots||x_q}$ is functional equivalent to any $(I_{\varphi,x'_1||\cdots||x'_q}$. If there exists any PPT adversary A that can compute x_i from the auxiliary input $(iO_r(I_{\varphi,x_1||\cdots||x_q}), i)$, there exists a PPT distinguisher D that can break the security of iO by invoking A. □

Remark.

1. Although the last assumption in Claim 3.2 looks quite strong, we were unable to contradict it.
2. If $\varphi \in L$ satisfies that for any PPT adversary B, it cannot determine whether $\varphi \in L_Y$ or $\varphi \in L_N$, then the claim does not need the third assumption. Since when $\varphi \in L_Y$, if there exists any PPT adversary A that can compute x_i from the auxiliary input $(iO_r(I_{\varphi,x_1||\cdots||x_q}), i)$, then there exists a PPT algorithm B that can determine whether $\varphi \in L_Y$ or $\varphi \in L_N$. $B(\varphi)$ invokes A with $(iO_r(I_{\varphi,x_1||\cdots||x_q}), i)$ and determines $\varphi \in L_Y$ if A outputs x_i, otherwise B determines $\varphi \in L_N$.

3.3 Construction of Public-coin diO for Multi-bit Hiding-input Point Function

Let O be an point obfuscator with auxiliary input for unpredictable distributions, and iO be an indistinguishability obfuscator for all poly-size circuits initiated by the construction in [18].

Given any $I_{\varphi,y}$ in the class of hiding-input point function with multi-bit output, where the length of y is n and $y = y_1, y_2, \cdots, y_n$, we first divide y into $q = q(\lambda)$ parts. And then we randomly choose $x_1, \cdots, x_q \in \{0,1\}^{\lambda}$ and compute an auxiliary input $aux = iO(I_{\varphi,x_1||\cdots||x_q})$. Thus that for all $i \in [q]$ we have $((aux, i), x_i) \in \mathcal{D}^i$, where $\mathcal{D}^i$ is as defined in section 3.2.

Next, based on the constant-self-composability of the AIPO scheme O for unpredictable distribution $\mathcal{D}^i$, we also use O to denote an obfuscation of constant-bit point function. Let c be any constant and $m = m_1, \cdots, m_c$, $O(I_{x_i,m})$ is defined as follows:

$$O(I_{x_i,m}) = O(I_{x_i}), O(I_{u_1}), \cdots, O(I_{u_c}),$$

where I_{x_i} and I_{u_j} ($j \in [c]$) are point functions, and for $j \in [c]$, $u_j = x_i$ if $m_j = 1$, and $u_j \in_R \{0,1\}^{\lambda}$ otherwise. The evaluation algorithm takes input $u \in \{0,1\}^{\lambda}$ and computes $O(I_{x_i})(u)$, and outputs $m = O(I_{u_1})(u), \cdots, O(I_{u_c})(u)$ if $O(I_{x_i})(u) = 1$. Otherwise, it outputs 0. Then for any $((aux, i), x_i) \in \mathcal{D}^i$ and for any $m, m' \in \{0,1\}^c$ we have

$$((aux, i), O(I_{x_i,m})) \stackrel{c}{=} ((aux, i), O(I_{x_i,m'})).$$

The indistinguishability follows from the $(c+1)$-self-composability of O.

Combined above results, we obtain a public-coin diO scheme for $I_{\varphi,y}$. We first choose a constant c and let

$$\left\{ \begin{array}{l} v_i = y_{(i-1)c+1}, \cdots, y_{ic},\ i \in [q-1] \\ v_q = y_{(q-1)c+1}, \cdots, y_n,\ i = q \end{array} \right. .$$

Then, we define

$$M_L = aux, O(I_{x_1,v_1}), \cdots, O(I_{x_q,v_q}).$$

On input (M_L, x), the evaluation algorithm $Evaluate_L$ first evaluates $aux(x) = iO(I_{\varphi,x_1||\cdots||x_q})(x)$ and outputs 0 if $aux(x) = 0$. Otherwise, it divides $aux(x)$ as $aux(x) = x_1, \cdots, x_q$ and outputs

$$y = O(I_{x_1,v_1})(x_1), \cdots, O(I_{x_q,v_q})(x_q).$$

Then, we obtain the public-coin diO algorithm for $I_{\varphi,y}$ as $diO_L(I_{\varphi,y}) = (M_L, Evaluate_L)$. The above construction is depicted in Figure 1.

1. Given a multi-bit hiding-input point function $I_{\varphi,y}$, choose a constant c and divide $y = y_1, y_2, \cdots, y_n$ into $q = q(\lambda)$ parts such that $(q-1)c < n \leq qc$,and for $i \in [q-1]$, $v_i = y_{(i-1)c+1}, \cdots, y_{ic}$ and $v_q = y_{(q-1)c+1}, \cdots, y_n$.
2. Uniformly select $x_1, \cdots, x_q \in \{0,1\}^\lambda$.
3. Uniformly select $r_0 \in \{0,1\}^{poly(\lambda)}$, compute $aux = iO(I_{\varphi,x_1||\cdots||x_q})$ with r_0.
4. For $i \in [q]$, uniformly select $r_i \in \{0,1\}^{poly(\lambda)}$, compute $O(I_{x_i,v_i})$ with r_i.
5. Set
$$M_L = aux, O(I_{x_1,v_1}), \cdots, O(I_{x_q,v_q}).$$
6. $Evaluate_L(M_L, x)$: First evaluate
$$aux(x) = iO(I_{\varphi,x_1||\cdots||x_q})(x)$$
and output 0 if $aux(x) = 0$.
Otherwise, divide $aux(x) = x_1, \cdots, x_q$ and output
$$y = O(I_{x_1,v_1})(x_1), \cdots, O(I_{x_q,v_q})(x_q).$$
7. Output $diO_L(I_{\varphi,y}) = (M_L, Evaluate_L)$.

Figure 1. Construction of public-coin diO for multi-bit hiding-input point function.

Theorem 3.1 *Assuming the construction $\mathcal{D}^i$ in section 3.2 is an unpredictable distribution and the existence of auxiliary input point obfuscation for unpredictable distributions, the construction in Figure 1 is a public-coin differing-inputs obfuscator for the class of hiding-input point function with multi-bit output.*

Proof. It is obvious that the construction in Figure 1 is a PPT algorithm and it keeps the functionality from $I_{\varphi,y}$, which can be seen from the evaluation algorithm.

Fix any public-coin differing-inputs sampler $Sam = \{Sam_\lambda\}$ for the class of hiding-input point function with multi-bit output $\{I_{\varphi,y}\}_{\varphi,y\in\{0,1\}^{n(\lambda)}}$, where $\varphi \in L$ (L is some language in NP) is a hard problem (see Definition 3.1). We need to prove that for any PPT distinguishing algorithm $\mathcal{A} = \{\mathcal{A}_\lambda\}$, there exists a negligible function $negl(\cdot)$ such that for all security parameters $\lambda \in N$:

$$\left|\Pr\left[\begin{array}{c}\mathcal{A}_\lambda(r,\widetilde{I}) = 1 : (I_{\varphi,y}, I_{\varphi,y'}) \leftarrow Sam_\lambda(r) \\ \widetilde{I} \leftarrow diO(1^\lambda, I_{\varphi,y})\end{array}\right]\right.$$
$$\left.-\Pr\left[\begin{array}{c}\mathcal{A}_\lambda(r,\widetilde{I}) = 1 : (I_{\varphi,y}, I_{\varphi,y'}) \leftarrow Sam_\lambda(r) \\ \widetilde{I} \leftarrow diO(1^\lambda, I_{\varphi,y'})\end{array}\right]\right| \leq negl(\lambda),$$

where the probability is taken over r and the coins of diO and $\mathcal{A}_\lambda$.

That is to prove that for any sampled hard problem φ, and for any $y, y' \in \{0,1\}^n$:

$$\{diO_L(I_{\varphi,y})\} \stackrel{c}{=} \{diO_L(I_{\varphi,y'})\}.$$

Let

$$\begin{cases} v_i = y_{(i-1)c+1}, \cdots, y_{ic},\ i \in [q-1] \\ v_q = y_{(q-1)c+1}, \cdots, y_n,\ i = q \end{cases}, \quad \begin{cases} v'_i = y'_{(i-1)c+1}, \cdots, y'_{ic},\ i \in [q-1] \\ v'_q = y'_{(q-1)c+1}, \cdots, y'_n,\ i = q \end{cases}.$$

Since the description of the evaluation algorithm is deterministic, for convenience, we do not write it in the obfuscation results in the following proof. For every $x_1, \cdots, x_q \in_R \{0,1\}^\lambda$, consider the following sequence of hybrid distributions. For $k \in \{0, 1, \cdots, q\}$, we define H_q^k:

$$H_q^k := \left\{ \left(iO(I_{\varphi, x_1||\cdots||x_q}), O(I_{x_1, v_1'}), \cdots, O(I_{x_k, v_k'}), O(I_{x_{k+1}, v_{k+1}}), \cdots, O(I_{x_q, v_q}) \right) \right\}.$$

Then H_q^0 is the distribution $\left\{ diO_L(I_{\varphi,y}) \right\}_{x_1,\cdots,x_q}$ and H_q^q is the distribution $\left\{ diO_L(I_{\varphi,y'}) \right\}_{x_1,\cdots,x_q}$.

To prove $\left\{ diO_L(I_{\varphi,y}) \right\} \stackrel{c}{=} \left\{ diO_L(I_{\varphi,y'}) \right\}$, we first prove that for every $x_1, \cdots, x_q \in_R \{0,1\}^\lambda$,

$$\left\{ diO_L(I_{\varphi,y}) \right\}_{x_1,\cdots,x_q} \stackrel{c}{=} \left\{ diO_L(I_{\varphi,y'}) \right\}_{x_1,\cdots,x_q},$$

which is to prove $H_q^0 \stackrel{c}{=} H_q^q$. And thus we need to prove that $H_q^k \stackrel{c}{=} H_q^{k+1}$ for every $k \in \{0, 1, \cdots, q-1\}$. Since $x_1, \cdots, x_q$ are independent, the q obfuscations are independent. Therefore, the indistinguishability between H_q^k and H_q^{k+1} follows from the indistinguishability between the distributions: $\left\{ \left(iO(I_{\varphi, x_1||\cdots||x_q}), O(I_{x_{k+1}, v_{k+1}}) \right) \right\}$ and $\left\{ \left(iO(I_{\varphi, x_1||\cdots||x_q}), O(I_{x_{k+1}, v_{k+1}'}) \right) \right\}$.

Since $\left\{ diO_L(I_{\varphi,y}) \right\} = \cup_{x_1,\cdots,x_q \in \{0,1\}^\lambda} \left\{ diO_L(I_{\varphi,y}) \right\}_{x_1,\cdots,x_q}$, then for any $\widetilde{I} \in \left\{ diO_L(I_{\varphi,y}) \right\}$, there would be some $x_1, \cdots, x_q \in \{0,1\}^\lambda$ such that $\widetilde{I} \in \left\{ diO_L(I_{\varphi,y}) \right\}_{x_1,\cdots,x_q}$. Then no PPT algorithm can distinguish $\widetilde{I}$ is an obfuscation of $I_{\varphi,y}$ or $I_{\varphi,y'}$.

This completes the proof. □

4 Public-coin-dependent Differing-inputs Obfuscation

Based on the results obtained in section 3, the goal in this section is to construct a public-coin-dependent differing-inputs obfuscator for any efficiently generated circuit pair $(C_0, C_1) \leftarrow Samp(r)$, where $Samp$ is any public-coin differing-inputs sampler for some parameterized collection of circuits $\mathcal{C}$ (see definition 2.5) and r is a random value. We first define an language $L = (L_Y, L_N)$ as follows:

$$\begin{cases} L_Y = \{(C_0, C_1) : \exists\, x \; such \; that \; C_0(x) \neq C_1(x)\} \\ L_N = \{(C_0, C_1) : C_0(x) = C_1(x) \; for \; all \; x\} \end{cases},$$

where $(C_0, C_1) \leftarrow Samp(r)$, $r \in_R \{0,1\}^{poly(\lambda)}$. Hence then, L is an NP language and we let $\mathcal{R}_L$ be the efficiently computable binary NP relation for L. Then, for any circuit pair $(C_0, C_1) \in L$, it is hard to find an input x such that $C_0(x) \neq C_1(x)$), then $\varphi = (C_0, C_1) \in L$ is a hard problem. And then we have that for any PPT algorithm A,

$$\Pr[\mathcal{R}_L(\varphi, w) = 1 : w \leftarrow A(\varphi)] < negl(\lambda).$$

In this condition, we can use the result in section 3 and hence then make use of the property of the circuit pair in the construction of public-coin-dependent diO.

Let $\mathcal{C} = \{\mathcal{C}_\lambda\}$ be a parameterized collection of polynomial-time circuits and $Samp$ is any public-coin differing-inputs sampler for $\mathcal{C}$. Let $FHE = (Gen, Enc, Dec, Eval)$ be a fully homomorphic encryption scheme. Let diO_L be a public-coin differing-inputs obfuscator for the class of function $\{I_{\varphi,y}\}_{\varphi \in L, y \in \{0,1\}^{n(\lambda)}}$, where L is defined as above. Let iO be a polynomially secure indistinguishability obfuscator for all poly-size circuits which is initiated by any polynomially secure indistinguishability obfuscator for NC^1 circuits and the construction in [18]. Let U_λ be an oblivious universal circuit which on input the description of a circuit B and a string x, executes B on x.

Different from the previous works on the construction of diO, our public-coin-dependent differing-inputs obfuscator takes as input the random coins r (used by the sampler) such that $\{C_0, C_1\} \leftarrow Samp(r)$ and obfuscates circuit $C \in \{C_0, C_1\}$, the one needed to be obfuscated. It means that besides the circuit to be obfuscated, the obfuscator needs to know another circuit such that it is hard to find an input on which the two circuits differ. Our construction is described by an obfuscation algorithm $Obfuscate$ and an evaluation algorithm $Evaluate$.

Obfuscator $Obfuscate(1^\lambda, C, r)$**:** Sample $C_0, C_1 \in \mathcal{C}_\lambda$ by $Samp(r)$ and check whether $C \in \{C_0, C_1\}$. If $C \in \{C_0, C_1\}$, proceed the following steps.

1. Generate two FHE key pairs $(pk_0, sk_0) \leftarrow Gen(1^\lambda)$, $(pk_1, sk_1) \leftarrow Gen(1^\lambda)$.
2. Generate ciphertexts $f_0 = Enc_{pk_0}(C)$, $f_1 = Enc_{pk_1}(C)$.
3. Let $\varphi = (C_0, C_1) \in L$. Compute $P_1 = diO_L(I_{\varphi, sk_0 || pk_0})$.
4. Generate an obfuscation of the decryption algorithm $Decrypt_0$ (see details in Figure 2) as $P_2 = iO(Decrypt_0)$.
5. The obfuscation components are output as: $C' = (P_1, P_2, pk_0, pk_1, f_0, f_1)$.

Evaluation $Evaluate(C' = (P_1, P_2, pk_0, pk_1, f_0, f_1), x)$**:** The $Evaluate$ algorithm takes in the obfuscation output C' and a circuit input x and computes the following.

1. Compute $a = P_1(x)$
2. If $a \neq 0$, divide it into $a = sk, pk$. Compute $C = Dec_{sk}(f_0)$ and output $C(x)$.
3. If $a = 0$, compute $ct_0 = Eval_{pk_0}(U_\lambda(\cdot, x), f_0)$ and $ct_1 = Eval_{pk_1}(U_\lambda(\cdot, x), f_1)$. Run $P_2(ct_0, ct_1, x)$ and output the result.

Then our public-coin-dependent diO algorithm is $diO(1^\lambda, C, r) = (Obfuscate(1^\lambda, C, r), Evaluate)$.

The decryption algorithm $Decrypt_0$ defined in Figure 2 only decrypts the ciphertexts meeting the conditions set in the algorithm. It is easy to see that the algorithms $Decrypt_0$ and $Decrypt_1$ (defined in Figure 3) are two equivalent circuits.

Decryption algorithm $Decrypt_0$**:**
Input: a tuple (ct_0, ct_1, x), **Constants:** $\varphi, pk_0, pk_1, f_0, f_1, sk_0$.
Proceeds as follows:

1. Check whether $\mathcal{R}_L(\varphi, x) = 1$, if the check succeeds, output $\perp$.
2. Else, check whether

$$ct_0 = Eval_{pk_0}(U_\lambda(\cdot, x), f_0) \wedge ct_1 = Eval_{pk_1}(U_\lambda(\cdot, x), f_1),$$

   if the check fails, output $\perp$.
3. Else, output $Dec_{sk_0}(ct_0)$.

Figure 2. The decryption algorithm $Decrypt_0$.

Decryption algorithm $Decrypt_1$**:**
Input: a tuple (ct_0, ct_1, x), **Constants:** $\varphi, pk_0, pk_1, f_0, f_1, sk_1$.
Proceeds as follows:

1. Check whether $\mathcal{R}_L(\varphi, x) = 1$, if the check succeeds, output $\perp$.
2. Else, check whether

$$ct_0 = Eval_{pk_0}(U_\lambda(\cdot, x), f_0) \wedge ct_1 = Eval_{pk_1}(U_\lambda(\cdot, x), f_1),$$

   if the check fails, output $\perp$.
3. Else, output $Dec_{sk_1}(ct_1)$.

Figure 3. The decryption algorithm $Decrypt_1$.

Remark. In the decryption algorithms $Decrypt_0$ and $Decrypt_1$, we use the FHE scheme with deterministic evaluation algorithm. In the case the decryption algorithms use the FHE scheme with probabilistic evaluation algorithm, the inputs of the decryption algorithms need to include the randomness used to compute ct_0, ct_1 with the probabilistic evaluation algorithm.

Theorem 4.1 *Assuming the construction in Figure 1 is a public-coin differing-inputs obfuscator for the class of hiding-input point function with multi-bit output and the existence of fully homomorphic encryption schemes. Then the construction above is a public-coin-dependent differing-inputs obfuscation.*

Proof. Since the algorithms used in the construction are all PPT algorithms, the obfuscation algorithm diO is also a PPT algorithm. The functionality follows from the correctness of the evaluation algorithm, which is guaranteed by the security of diO_L algorithm and the correctness of the FHE algorithm.

To begin the proof of security, we first recall the security of the diO_L algorithm for $\{I_{\varphi,y}\}_{\varphi\in L, y\in_R\{0,1\}^n}$, where L is defined in the beginning of section 4. Fix any public-coin differing-inputs sampler $Sam = \{Sam_\lambda\}$, for any PPT distinguishing algorithm $\mathcal{A} = \{\mathcal{A}_\lambda\}$, there exists a negligible function $negl(\cdot)$ such that for all security parameters $\lambda \in N$:

$$\left|\Pr\left[\begin{array}{c}\mathcal{A}_\lambda(r,\widetilde{I})=1:(I_{\varphi,y},I_{\varphi,y'})\leftarrow Sam_\lambda(r)\\ \widetilde{I}\leftarrow diO(1^\lambda,I_{\varphi,y})\end{array}\right]\right.$$
$$\left.-\Pr\left[\begin{array}{c}\mathcal{A}_\lambda(r,\widetilde{I})=1:(I_{\varphi,y},I_{\varphi,y'})\leftarrow Sam_\lambda(r)\\ \widetilde{I}\leftarrow diO(1^\lambda,I_{\varphi,y'})\end{array}\right]\right| \leq negl(\lambda),$$

where the probability is taken over r and the coins of diO and $\mathcal{A}_\lambda$.

Since the description of the evaluation algorithm is deterministic, for convenience, we do not write it in the obfuscation results in the following proof. Now we assume for the contradiction that there exists a public-coin differing-inputs sampler $Samp$, a PPT adversary $\mathcal{T}$, a polynomial $p = p(\lambda)$ and a security parameters $\lambda \in N$ such that

$$\left|\Pr\left[\begin{array}{c}\mathcal{T}_\lambda(r,C')=1:(C_0,C_1)\leftarrow Samp_\lambda(r),\\ C'\leftarrow diO(1^\lambda,C_0)\end{array}\right]\right.$$
$$\left.-\Pr\left[\begin{array}{c}\mathcal{T}_\lambda(r,C')=1:(C_0,C_1)\leftarrow Samp_\lambda(r),\\ C'\leftarrow diO(1^\lambda,C_1)\end{array}\right]\right| > 1/p,$$

where the probability is taken over r and the coins of diO and $\mathcal{T}_\lambda$. Then, we construct a PPT adversary $A_\lambda(\cdot,\cdot)$ to break the security of diO_L algorithm for $\{I_{\varphi,y}\}_{\varphi\in L, y\in_R\{0,1\}^n}$, where n is the length of the key pair of the FHE scheme, $(C_0, C_1) \leftarrow Samp_\lambda(r)$, and $\varphi = (C_0, C_1) \in L$.

We define a public-coin differing-inputs sampler $Sam = \{Sam_\lambda\}$ for $\{I_{\varphi,y}\}_{\varphi\in L, y\in_R\{0,1\}^n}$ as follows: On input $(r, r_0, r_1) \in \{0,1\}^{poly(\lambda)}$

1. Sample $C_0, C_1 \in \mathcal{C}_\lambda$ by $Samp_\lambda(r)$ and let $\varphi = (C_0, C_1) \in L$.
2. Generate FHE key pairs $(pk^0, sk^0) \leftarrow Gen_{r_0}(1^\lambda)$, $(pk^1, sk^1) \leftarrow Gen_{r_1}(1^\lambda)$,
3. Output $(I_{\varphi,sk^0||pk^0}, I_{\varphi,sk^1||pk^1})$.

Then we construct a PPT adversary $A_\lambda(\cdot,\cdot)$ as follows:
Adversary $A((r, r_0, r_1), diO_L(I_{\varphi,y}))$**:** Sample $(I_{\varphi,sk^0||pk^0}, I_{\varphi,sk^1||pk^1})$ by $Sam_\lambda(r, r_0, r_1)$ and obtain $\varphi = (C_0, C_1)$, FHE key pairs $(pk^0, sk^0), (pk^1, sk^1)$, and set $P_1 = diO_L(I_{\varphi,y})$.

1. Select $r_2 \in \{0,1\}^{poly(\lambda)}$, generate FHE key pair $(pk_1, sk_1) \leftarrow Gen_{r_2}(1^\lambda)$.
2. Select $b \in_R \{0,1\}$.
3. Compute $f_0 = Enc_{pk^b}(C_b)$, $f_1 = Enc_{pk_1}(C_b)$.
4. Select $r_4 \in \{0,1\}^{poly(\lambda)}$ and compute $P_2 = iO(Decrypt_0; r_4)$ with constants φ, $pk_0 = pk^b$, pk_1, f_0, f_1, $sk_0 = sk^b$.
5. Invoke $\mathcal{T}(r,\cdot)$ with $C' = (P_1, P_2, pk^b, pk_1, f_0, f_1)$ and output $b' = \mathcal{T}(r, C')$.

Now we compute the probability that A succeeds in distinguishing $y = sk^0||pk^0$ or $y = sk^1||pk^1$. For convenience in proving, we assume that $y = sk^0||pk^0$.

$$\begin{aligned}\Pr[b'=0] &= \Pr[b'=b\wedge b=0]+\Pr[b'=1-b\wedge b=1]\\ &= \frac{1}{2}\Pr[b'=0|b=0]+\frac{1}{2}\Pr[b'=0|b=1]\\ &= \frac{1}{2}\Pr[\mathcal{T}succeeds|b=0]+\frac{1}{2}\Pr[\mathcal{T}fails|b=1],\end{aligned}$$

$$\begin{aligned}\Pr[b'=1] &= \Pr[b'=1-b\wedge b=0]+\Pr[b'=b\wedge b=1]\\ &= \frac{1}{2}\Pr[b'=1|b=0]+\frac{1}{2}\Pr[b'=1|b=1]\\ &= \frac{1}{2}\Pr[\mathcal{T}fails|b=0]+\frac{1}{2}\Pr[\mathcal{T}succeeds|b=1].\end{aligned}$$

We claim that $(P_2, pk^b, pk_1, f_0, f_1)$ with the ciphertexts corresponding to C_0 and $(P_2, pk^b, pk_1, f_0, f_1)$ with the ciphertexts corresponding to C_1 are indistinguishable. The proof with a sequence of hybrids follows from the proof of security of the poly-sized circuit iO algorithm in [18]. The indistinguishability is proved from the IND-CPA property of the PKE scheme (included in the FHE scheme) and the indistinguishability security of the obfuscator for all poly-size circuits. Thus we can see that for a random $y^* \in \{0,1\}^n \backslash \{sk^b||pk^b, sk_1||pk_1\}$, $(diO_L(I_{\varphi,y^*}), P_2, pk^b, pk_1, f_0, f_1)$ with the ciphertexts corresponding to C_0 and $(diO_L(I_{\varphi,y^*}), P_2, pk^b, pk_1, f_0, f_1)$ with the ciphertexts corresponding to C_1 are indistinguishable. Otherwise, we can obtain an adversary for $(P_2, pk^b, pk_1, f_0, f_1)$. It just chooses a random $y^* \in \{0,1\}^n$, computes $diO_L(I_{\varphi,y^*})$, and invokes the adversary for $(diO_L(I_{\varphi,y^*}), P_2, pk^b, pk_1, f_0, f_1)$ to break the indistinguishability. Hence then, in the condition that $y = sk^0||pk^0$, we have $\Pr[\mathcal{T}succeeds|b=1] - \Pr[\mathcal{T}fails|b=1] < negl(\lambda)$. And then we have

$$\begin{aligned}
&\Pr[b'=0] - \Pr[b'=1] \\
&= \frac{1}{2}(\Pr[\mathcal{T}succeeds|b=0] - \Pr[\mathcal{T}fails|b=0]) + \frac{1}{2}(\Pr[\mathcal{T}fails|b=1] - \Pr[\mathcal{T}succeeds|b=1]) \\
&> \frac{1}{2p} - negl(\lambda).
\end{aligned}$$

This contradicts the security of diO_L algorithm for the function class $\{I_{\varphi,y}\}_{\varphi \in L, y \in_R \{0,1\}^n}$. Hence then, this completes the proof. □

5 Applications

Essentially, our public-coin-dependent diO algorithm works once it obtains the hard problem $\varphi = (C_0, C_1) \in L$ instead of the random coins r used to sample (C_0, C_1). To the best of our knowledge, in most application scenarios of diO and iO, there would be two special defined circuits satisfying the working conditions of diO and iO (respectively), such as the functional encryption scheme for circuits constructed from iO [18], the functional encryption scheme for Turing Machines constructed from public-coin diO [23], the public-key encryption scheme from iO [27], the four message concurrent zero knowledge protocol from public-coin diO [26], and the constant-round concurrent zero knowledge protocol from iO [15]. Hence then, besides the circuit to be obfuscated, it is reasonable to require the obfuscator to know another circuit such that it is hard to find an input on which the two circuits differ. In the followings, we show the applications of our constructions. We first use our public-coin diO for multi-bit hiding-input point functions to construct witness encryption schemes. And then we show that our public-coin-dependent diO algorithm can be applied to public-key encryption schemes.

5.1 Witness Encryption.

Definition 5.1 ***Witness Encryption [20].*** *A witness encryption scheme for an NP language $L = (L_Y, L_N)$ (with corresponding witness relation $\mathcal{R}$) consists of the following two polynomial-time algorithms:*

> ***Encryption.*** *The algorithm $Encrypt(1^\lambda, \varphi, m)$ takes as input a security parameter 1^λ, an unbounded-length string φ, and a message $m \in M$ for some message space M, and outputs a ciphertext CT.*
> ***Decryption.*** *The algorithm $Decrypt(CT, w)$ takes as input a ciphertext CT and an unbounded-length string w, and outputs a message m or the symbol $\perp$.*

These algorithms satisfy the following two conditions:

> ***Correctness.*** *For any security parameter λ, for any $m \in M$, and for any $\varphi \in L_Y$ such that $\mathcal{R}(\varphi, w)$ holds, there exists a negligible function $negl(\cdot)$, such that:*
>
> $$\Pr[Decrypt(Encrypt(1^\lambda, \varphi, m), w) = m] = 1 - negl(\lambda).$$
>
> ***Soundness Security.*** *For any $\varphi \in L_N$, for any PPT adversary A and messages $m_0, m_1 \in M$, there exists a negligible function $negl(\cdot)$, such that:*
>
> $$\left|\Pr[A(Encrypt(1^\lambda, \varphi, m_0)) = 1] - \Pr[A(Encrypt(1^\lambda, \varphi, m_1)) = 1]\right| < negl(\lambda).$$

Witness encryption does not require any setup algorithm.

We additionally define message indistinguishability for a witness encryption scheme, which implies the soundness security:

Message Indistinguishability. For any $\varphi \in L$, for any PPT adversary A and messages $m_0, m_1 \in M$, there exists a negligible function $negl(\cdot)$, such that:

$$\left|\Pr[A(Encrypt(1^\lambda, \varphi, m_0)) = 1] - \Pr[A(Encrypt(1^\lambda, \varphi, m_1)) = 1]\right| < negl(\lambda).$$

We show a witness encryption scheme with correctness and message indistinguishability for the NP language $L = (L_Y, L_N)$ as follows:

$$\begin{cases} L_Y = \{(C_0, C_1) : \exists\, x \; such \; that \; C_0(x) \neq C_1(x)\} \\ L_N = \{(C_0, C_1) : C_0(x) = C_1(x) \; for \; all \; x\} \end{cases},$$

where $(C_0, C_1) \leftarrow Samp(r)$, $r \in_R \{0,1\}^{poly(\lambda)}$ and $Samp$ is any public-coin differing-inputs sampler for some parameterized collection of circuits $\mathcal{C}$. Let diO_L be our public-coin diO for multi-bit hiding-input point functions.

Construction 5.1. Witness encryption scheme $Encrypt(1^\lambda, \varphi, \cdot, \cdot), Decrypt(1^\lambda, \cdot, w)$.
$Encrypt(1^\lambda, \varphi, \cdot, \cdot)$: On input a message $m \in \{0,1\}^\lambda$, it chooses $r \in \{0,1\}^{poly(\lambda)}$ and computes

$$CT = Encrypt(1^\lambda, \varphi, m, r) = diO_L(I_{\varphi,m}; r).$$

$Decrypt(1^\lambda, \cdot, w)$: On input a ciphertext CT, it runs CT on w and returns $m = CT(w)$.

The correctness of decryption is immediate. Note that for any $x \in L$, for any messages $m_0, m_1 \in M$, the indistinguishability between $Encrypt(1^\lambda, \varphi, m_0)$ and $Encrypt(1^\lambda, \varphi, m_1)$ follows from the indistinguishability between $diO_L(I_{\varphi,m_0})$ and $diO_L(I_{\varphi,m_1})$.

5.2 Public-key Encryption.

To obtain public-key encryption scheme from our public-coin-dependent diO algorithm, we first recall the IND-CPA secure public-key encryption (PKE) scheme in [27]. The PKE scheme is transferred from a secret key encryption (SKE) scheme by producing public key as an obfuscation of the SKE's encryption algorithm. The encryption algorithm makes uses of iO, pseudorandom generator (PRG), and puncturable pseudorandom functions. It is obvious that a public-coin diO scheme can be applied to the PKE scheme. Now we show that our public-coin-dependent diO algorithm can also be applied to the PKE scheme.

Definition 5.2 ***Puncturable Pseudorandom Functions [27].*** *A puncturable family of PRFs F mapping is given by a triple of Turing Machines Key_F, $Puncture_F$, and $Eval_F$, and a pair of computable functions $n(\cdot)$ and $m(\cdot)$, satisfying the following conditions:*

- ***[Functionality preserved under puncturing]*** *For every PPT adversary A such that $A(1^\lambda)$ outputs a set $S \subseteq \{0,1\}^{n(\lambda)}$, then for all $x \in \{0,1\}^{n(\lambda)}$ where $x \notin S$, we have that:*

$$\Pr[Eval_F(K, x) = Eval_F(K_S, x) : K \leftarrow Key_F(1^\lambda), K_S = Puncture_F(K, S)] = 1.$$

- ***[Pseudorandom at punctured points]*** *For every PPT adversary (A_1, A_2) such that $A_1(1^\lambda)$ outputs a set $S \subseteq \{0,1\}^{n(\lambda)}$ and state σ, consider an experiment where $K \leftarrow Key_F(1^\lambda)$ and $K_S = Puncture_F(K, S)$. Then we have*

$$\left|\Pr[A_2(\sigma, K_S, S, Eval_F(K, S)) = 1] - \Pr[A_2(\sigma, K_S, S, U_{m(\lambda)\cdot|S|}) = 1]\right| = negl(\lambda),$$

where $Eval_F(K, S)$ denotes the concatenation of $Eval_F(K, x_1)), \cdots, Eval_F(K, x_k))$ where $S = \{x_1, \cdots, x_k\}$ is the enumeration of the elements of S in lexicographic order, $negl(\cdot)$ is a negligible function, and U_l denotes the uniform distribution over l bits.

For ease of notation, we write $F(K, x)$ to represent $Eval_F(K, x)$. We also represent the punctured key $Puncture_F(K, S)$ by $K(S)$.

Let $G : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ be a PRG. Let $F : \{0,1\}^{2\lambda} \to \{0,1\}^l$ be a puncturable PRF. Let diO be a public-coin-dependent diO algorithm as described in subsection 4.1. And we define a public-coin differing-inputs sampler $Samp_\lambda(\cdot)$ such that on input a random value $r_d = (K, t^*)$, $Samp_\lambda(r_d)$ outputs two algorithms as described in Figure 5 and Figure 6. The scheme is defined as $PKE = (Gen, Enc, Dec)$.

- **Key generation** $Gen(1^\lambda)$**:** It first chooses a puncturable PRF key K for F. And then it chooses $t^* \in \{0,1\}^{2\lambda}$ at random, and then invokes $Samp_\lambda(K, t^*)$ to produce the algorithms of Figure 5 and Figure 6. Next, it creates an obfuscation of the $Encrypt$ algorithm of Figure 5 with diO. The size of $Encrypt$ is padded to be the maximum of itself and $Encrypt^*$ of Figure 6. The public key pk is the obfuscated algorithm $diO(Encrypt, (K, t^*))$ and the secret key $sk = K$.
- **Encryption** $Enc(pk, m \in \{0,1\}^l)$**:** It chooses a random value $r \in \{0,1\}^\lambda$ and runs pk on inputs m, r.
- **Decryption** $Dec(sk, c = (c_1, c_2))$**:** It outputs $m' = F(K, c_1) \oplus c_2$.

Constants: Punctured PRF key K.
Inputs: message $m \in \{0,1\}^l$,
randomness $r \in \{0,1\}^\lambda$.
$Encrypt$ proceeds as follows:

1. Compute $t = G(r)$.
2. Output $c = (c_1 = t, c_2 = F(K, t) \oplus m)$.

Figure 5. Encryption algorithm $Encrypt$.

Constants: Punctured PRF key $K(\{t^*\})$.
Inputs: message $m \in \{0,1\}^l$,
randomness $r \in \{0,1\}^\lambda$.
$Encrypt^*$ proceeds as follows:

1. Compute $t = G(r)$.
2. Output $c = (c_1 = t, c_2 = F(K, t) \oplus m)$.

Figure 6. Encryption algorithm $Encrypt^*$.

Theorem 5.1 *If our obfuscation scheme diO is secure as defined, G is a secure PRG, and F is a secure puncturable PRF, then our encryption scheme is IND-CPA secure.*

The proof of security follows from the proof of the PKE scheme in the early version of [27] with the modifications that the obfuscation algorithm is replaced with our public-coin-dependent diO algorithm. We give the following proof sketch.

Proof. To prove the security, we first describe a sequence of hybrid experiments.

- H_0: In the first hybrid the original IND-CPA security game is played.
  1. K is chosen as a key for F. $t^* \in \{0,1\}^{2\lambda}$ is chosen at random. The challenger then invokes $Samp_\lambda(K, t^*)$.
  2. The public key $pk = diO(Encrypt, rd = (K, t^*))$.
  3. The attacker receives pk and then gives $m_0, m_1 \in \{0,1\}^l$ to the challenger.
  4. The challenger randomly chosen $b \in \{0,1\}$, $r \in \{0,1\}^\lambda$, and compute the challenge ciphertext $c^b = (c_1, c_2)$, where $c_1 = t = G(r), c_2 = F(K, t) \oplus m_b$.
  5. The attacker receives the challenge ciphertext and guesses the bit b.
- H_1: It is the same as H_0 except that in step 4 t is chosen randomly in $\{0,1\}^{2\lambda}$. Since r is not in the attacker's view, it does not need to be generated.
- H_2: It is the same as H_1 except that in step 4 the challenger sets $t = t^*$.
- H_3: It is the same as H_2 except that in step 2 the public key $pk = diO(Encrypt^*, rd = (K, t^*))$.
- H_4: It is the same as H_3 except that the challenge ciphertext is $c^b = (c_1 = t^*, c_2 = z^*)$, where $z^* \in \{0,1\}^l$ is a random value.

Now we show that for $i \in \{0, 1, 2, 3\}$, H_i and H_{i+1} are indistinguishable. Hence then, any poly-time attacker's advantages in H_i and H_{i+1} are the same except for a negligible probability.

Step1: $H_0 \stackrel{c}{=} H_1$. This follows from the security of the pseudorandom generator G. If there exists any poly-time attacker A that can distinguish H_1 and H_2, we can obtain a PPT distinguisher D breaks the security of the pseudorandom generator G. Once receives a PRG challenge t, D runs the IND-CPA security game and sets $c^b = (t, F(K, t) \oplus m_b)$, if t is an output of the PRG G, it is H_0, and if t is chosen at random, it is H_1. D invokes A and outputs 0 (t is an output of the PRG) if A outputs H_0, and D outputs 1 (t is chosen at random) if A outputs H_1.

Step2: $H_1 \stackrel{c}{=} H_2$. This follows from the security of the indistinguishability obfuscator for all poly-size circuits. We first show that $diO(Encrypt, (K, t^*)) \stackrel{c}{=} diO(Encrypt, (K, t))$. Since the description of the evaluation algorithm is deterministic, for convenience, we do not write it in the obfuscation results in the following proof.

$diO(Encrypt, (K, t^*))$ is computed as follows: Sample $C_0 = Encrypt, C_1 = Encrypt^*$ by $Samp(r_d = (K, t^*))$ and check whether $Encrypt \in \{C_0, C_1\}$. If $Encrypt \in \{C_0, C_1\}$, proceed the following steps.

1. Generate two FHE key pairs $(pk_0, sk_0) \leftarrow Gen(1^\lambda)$, $(pk_1, sk_1) \leftarrow Gen(1^\lambda)$.
2. Generate ciphertexts $f_0 = Enc_{pk_0}(Encrypt)$, $f_1 = Enc_{pk_1}(Encrypt)$.
3. Let $\varphi_1 = (C_0, C_1) \in L$. Compute $P_1 = diO_L(I_{\varphi_1, sk_0||pk_0})$.
4. Generate an obfuscation as $P_2 = iO(Decrypt_0^1)$, where $Decrypt_0^1$ is the decryption algorithm (with constants φ_1, pk_0, pk_1, f_0, f_1, sk_0) defined in Figure 2.
5. Output $C_1' = (P_1, P_2, pk_0, pk_1, f_0, f_1)$.

$diO(Encrypt, (K, t))$ is computed as follows: Sample $C_2 = Encrypt, C_3 = Encrypt^*$ by $Samp(r_d = (K, t))$ and check whether $Encrypt \in \{C_2, C_3\}$. If $Encrypt \in \{C_2, C_3\}$, proceed the following steps only when the check succeeds.

1. Generate two FHE key pairs $(pk_0, sk_0) \leftarrow Gen(1^\lambda)$, $(pk_1, sk_1) \leftarrow Gen(1^\lambda)$.
2. Generate ciphertexts $f_0 = Enc_{pk_0}(Encrypt)$, $f_1 = Enc_{pk_1}(Encrypt)$.
3. Let $\varphi_2 = (C_2, C_3) \in L$. Compute $P_3 = diO_L(I_{\varphi_2, sk_0||pk_0})$.
4. Generate an obfuscation as $P_4 = iO(Decrypt_0^2)$, where $Decrypt_0^2$ is the decryption algorithm (with constants φ_2, pk_0, pk_1, f_0, f_1, sk_0) defined in Figure 2.
5. Output $C_2' = (P_3, P_4, pk_0, pk_1, f_0, f_1)$.

We first observe that since t^* and t are chosen at random, with probability $(1 - \frac{1}{2^\lambda})^2$, t^* and t are not in the image on the PRG algorithm G. Then, with all but negligible probability that the input/output behaviors of $I_{\varphi_1, sk_0||pk_0}$ and $I_{\varphi_2, sk_0||pk_0}$ are identical and the input/output behaviors of $Decrypt_0^1$ and $Decrypt_0^2$ are identical. Then, the indistinguishability between $(diO_L(I_{\varphi_1, sk_0||pk_0}), iO(Decrypt_0^1))$ and $(diO_L(I_{\varphi_2, sk_0||pk_0}), iO(Decrypt_0^2))$ is guaranteed by the security of the indistinguishability obfuscator used in diO_L and the security of iO. That is

$$(diO_L(I_{\varphi_1, sk_0||pk_0}), iO(Decrypt_0^1)) \stackrel{c}{=} (diO_L(I_{\varphi_2, sk_0||pk_0}), iO(Decrypt_0^1)),$$

and

$$(diO_L(I_{\varphi_2, sk_0||pk_0}), iO(Decrypt_0^1)) \stackrel{c}{=} (diO_L(I_{\varphi_2, sk_0||pk_0}), iO(Decrypt_0^2)).$$

Now, since C_0 and C_2 are the same as $Encrypt$ in Figure 5, $diO(Encrypt, (K, t^*)) \stackrel{c}{=} diO(Encrypt, (K, t))$. Then, if there exists any poly-time attacker A that can distinguish H_1 and H_2, we can obtain a PPT distinguisher $D(K, t^*, t)$ that can distinguish $diO(Encrypt, (K, t^*))$ and $diO(Encrypt, (K, t))$. Once receives an obfuscation, $D(K, t^*, t)$ sets pk to be the obfuscation. When $D(K, t^*, t)$ receives the challenge message (m_0, m_1), it chooses $b \in \{0, 1\}$ at random and compute challenge ciphertext $c^b = (t^*, F(K, t^*) \oplus m_b)$. If the obfuscation is $diO(Encrypt, (K, t^*))$, it is H_2. If the obfuscation is $diO(Encrypt, (K, t))$, it is H_1. Then, $D(K, t^*, t)$ can distinguish $diO(Encrypt, (K, t^*))$ and $diO(Encrypt, (K, t))$ by invoking A with $(pk, (m_0, m_1), c^b)$ and outputs 0 (the obfuscation is $diO(Encrypt, (K, t))$) if A outputs H_1, and $D(K, t^*, t)$ outputs 1 (the obfuscation is $diO(Encrypt, (K, t^*))$) if A outputs H_2.

Step3: $H_2 \stackrel{c}{=} H_3$. This follows from the security of the public-coin-dependent diO algorithm diO. If there exists any poly-time attacker A that can distinguish H_2 and H_3, we can obtain a PPT distinguisher $D(K, t^*)$ that can distinguish $diO(Encrypt, (K, t^*))$ and $diO(Encrypt^*, (K, t^*))$. Once receives an obfuscation, $D(K, t^*)$ sets pk to be the obfuscation. When $D(K, t^*)$ receives the challenge message (m_0, m_1), it chooses $b \in \{0, 1\}$ at random and compute challenge ciphertext $c^b = (t^*, F(K, t^*) \oplus m_b)$. If the obfuscation is $diO(Encrypt, (K, t^*))$, it is H_2. If the obfuscation is $diO(Encrypt^*, (K, t^*))$, it is H_3. Then, $D(K, t^*)$ can distinguish $diO(Encrypt, (K, t^*))$ and $diO(Encrypt^*, (K, t^*))$ by invoking A with $(pk, (m_0, m_1), c^b)$, and then $D(K, t^*)$ outputs 0 (the obfuscation is $diO(Encrypt, (K, t^*))$) if A outputs H_2, and $D(K, t^*)$ outputs 1 (the obfuscation is $diO(Encrypt^*, (K, t^*))$) if A outputs H_3.

Step4: $H_3 \stackrel{c}{=} H_4$. This follows from the security of the puncturable PRF F. If there exists any poly-time attacker A that can distinguish H_3 and H_4, we can obtain a PPT distinguisher D breaks the selective security of the constrained pseudorandom function at the punctured points. D runs the security game of H_3, except that it gets the punctured PRF key $K(\{t^*\})$ and challenge a. It continues to run as in H_3 except that it creates the challenge ciphertext as $c^b = (t^*, a \oplus m_b)$. If a is the output of the PRF at point t^*, then we are in H_3. If it was chosen uniformly at random, we are in H_4. D invokes A and outputs 0 (a is an output of the PRF at point t^*) if A outputs H_3, and D outputs 1 (a is chosen at random) if A outputs H_4.

Then the advantage of any poly-time attacker in H_4 is 0, since it conveys no information about the bit b. This completes the proof of the PKE scheme's IND-CPA security. □

6 Conclusion

In this paper, we present a new construction of public-coin-dependent diO based on the existence of iO, point obfuscation with auxiliary input (AIPO) and fully homomorphic encryption scheme (FHE). Instead of assuming the existence of public-coin differing-inputs obfuscator for circuits in the class NC^1, we first construct a public-coin differing-inputs obfuscator for the class of multi-bit hiding-input point function with iO and AIPO, and then use it to complete the public-coin-dependent diO for any pair of circuits that are hard to be found an input on which their outputs differ. We claim that public-coin-dependent diO for circuits can be applied to the constructions of important cryptographic primitives and protocols.

AIPO can be implied by Canetti's strong DDH assumption [13] if it does not contradicts the existence if iO. The assumption essentially states that there exists an ensemble of prime order groups $\mathcal{G} = \{G_\lambda : |G_\lambda| = p_\lambda, |p_\lambda| = \lambda+1\}$ such that for any unpredictable distribution $D = \{D_\lambda = (Z_\lambda, X_\lambda)\}_{\lambda \in N}$ with support $\{0,1\}^{poly(\lambda)} \times Z_{p_\lambda}$, it holds that $(z, g, g^x) \stackrel{c}{=} (z, g, g^u)$, where $(z, x) \leftarrow (Z_\lambda, X_\lambda)$, $u \in_R Z_{p_\lambda}$ and g is a random generator of G_λ.

We leave it an open question that whether a public-coin differing-inputs obfuscator for the class of multi-bit hiding-input point function can be achieved based on some weaker assumptions.

References

1. Ananth, P., Boneh, D., Garg, S., Sahai, A., Zhandry, M.: Differing-inputs obfuscation and applications. IACR Cryptology ePrint Archive, 2013
2. Barak, B., Garg, S., Kalai, Y.T.: Protecting obfuscation against algebraic attacks. In: Nguyen, P.Q., Oswald, E. (eds.): EUROCRYPT 2014, LNCS 8441, pp. 221-238, 2014
3. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1-18. Springer, Heidelberg (2001)
4. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. J. ACM 59(2), 6 (2012)
5. Bellare, M., Stepanovs, I., Tessaro, S.: Poly-many hardcore bits for any one-way function and a framework for differing-inputs obfuscation. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part II. pp. 102-121
6. Bellare, M., Stepanovs, I., Waters, B.: New negative results on differing-inputs obfuscation. In: Fischlin, M. and Coron, J.-S. (eds.) EUROCRYPT 2016, Part II, LNCS 9666, pp. 792-821, 2016
7. Bitansky, N., Canetti, R.: On Strong simulation and composable point obfuscation. J. Cryptology 27(2): 317-357 (2014)
8. Bitansky, N., Paneth, O.: Point obfuscation and 3-round zero-knowledge. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 190-208. Springer, Heidelberg (2012)
9. Boyle, E., Pass, R.: Limits of extractability assumptions with distributional auxiliary input. In ASIACRYPT 2015, Part II, LNCS 9453, pp. 236-261, 2015
10. Brakerski, Z., Dagmi, O.: Shorter circuit obfuscation in challenge security models. In: Zikas, V., Prisco, R.D. (eds.): SCN 2016, LNCS 9841, pp. 551-570, 2016
11. Brakerski, Z., Rothblum, G.N.: Virtual black-box obfuscation for all circuits via generic graded encoding. In: Lindell, Y. (ed.): TCC 2014, LNCS 8349, pp. 1-25, 2014
12. Brzuska, C., Mittelbach, A.: Indistinguishability obfuscation versus point obfuscation with auxiliary input. In ASIACRYPT 2014, LNCS, vol. 8874, pp. 142-161, 2014
13. Canetti, R.: Towards realizing random oracles: Hash functions that hide all partial information. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 455-469. Springer, Heidelberg (1997)
14. Canetti, R., Dakdouk, R.R.: Obfuscating point functions with multibit output. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 489-508. Springer, Heidelberg (2008)
15. Chung, K.-M., Lin, H., Pass, R.: Constant-round concurrent zero-knowledge from indistinguishability obfuscation. In: CRYPTO 2015, Part I, pp. 287-307
16. Canetti, R., Tauman Kalai, Y., Varia, M., Wichs, D.: On symmetric encryption and point obfuscation. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 52-71. Springer, Heidelberg (2010)
17. Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 1-17. Springer, Heidelberg (2013)
18. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: FOCS, pp. 40-49 (2013)
19. Garg, S., Gentry, C., Halevi, S., Wichs, D.: On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. pp. 518-535
20. Garg, S., Gentry, C., Sahai, A., Waters, B.: Witness encryption and its applications. In STOC, 2013.
21. Goldwasser, S., Kalai, Y.-T.: On the impossibility of obfuscation with auxiliary input. FOCS 2005: 553-562

22. Garg, S., Miles, E., Mukherjee, P., Sahai, A., Srinivasan, A., Zhandry, M.: Secure obfuscation in a weak multilinear map model. In: Hirt, M., Smith, A. (eds.): TCC 2016-B, Part II, LNCS 9986, pp. 241-268, 2016.
23. Ishai, Y., Pandey, O., Sahai, A.: Public-coin differing-inputs obfuscation and its applications. In Dodis, Y. and Nielsen, J.B. (eds.) TCC 2015, Part II, LNCS 9015, pp. 668-697, 2015
24. Lynn, B., Prabhakaran, M., Sahai, A.: Positive results and techniques for obfuscation. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 20-39. Springer, Heidelberg (2004)
25. Miles, E., Sahai, A., Zhandry, M.: Annihilation Attacks for Multilinear Maps: Cryptanalysis of Indistinguishability Obfuscation over GGH13. In: Robshaw, M., Katz, J. (eds.): CRYPTO 2016, Part II, LNCS 9815, pp. 629-658, 2016
26. Pandey, O., Prabhakaran, M., Sahai, A.: Obfuscation-based non-black-box simulation and four message concurrent zero knowledge for NP. In: TCC (2015), Earlier version: IACR Cryptology ePrint Archive 2013:754
27. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: STOC, 2014. Earlier version: Cryptology ePrint Archive, Report 2013/454, 2013. http://eprint.iacr.org/.
28. Wee, H.: On obfuscating point functions. STOC 2005: 523-532
29. Yupu Hu, Huiwen Jia.: Cryptanalysis of GGH map. In: EUROCRYPT 2016, Part I, LNCS 9665, pp. 537-565.
30. Zimmerman, J.: How to obfuscate programs directly. In: EUROCRYPT 2015, Part II, LNCS 9057, pp. 439-467.