A Secure Oblivious Transfer Protocol from Indistinguishing Obfuscation*

Mei Wang[1,2] Zheng Yuan[1,2]**Xiao Feng[1,2]

[1] Beijing Electronic Science & Technology Institute, Beijing 100070, P. R. China
zyuan@tsinghua.edu.cn and sxzyfx@163.com
[2] School of Telecommunications Engineering, Xidian University, Shaanxi 710071,P. R. China

Abstract. We proposed a new secure oblivious transfer protocol from indistinguishability obfuscation in this paper. Our main technical tool is the candidate indistinguishability obfuscation introduced in [1] and a dual-mode cryptosystem proposed in [2]. Following their steps, we presents a new k-out-of-l oblivious transfer protocol, its realization from DDH is described in this paper, in which we combined indistinguishability obfuscation with the dual-mode cryptosystem. The security of our scheme mainly relies on the indistinguishability of the obf-branches (corresponding to the two modes in dual-mode model). Our paper explores a new way for the application of indistinguishability obfuscation.

Keywords: Indistinguishability obfuscation, Oblivious transfer, dual-mode cryptosystem.

1 Introduction

Oblivious Transfer(OT) is a two-party protocol, which was first proposed by Rabin[3] in 1981 and developed by Even, Goldreich and Lempel[4]. Oblivious Transfer is an important tool to construct cryptographic primitives and has been widely used in various applications such as signing fair electronic contract, private information retrieval and secure multi-party computation and so on. Oblivious Transfer is such a protocol which allows receiver obtain exactly one of several values from sender. The receiver don't know other values except the one he chose. The sender don't know which value was received. There are a lot of cryptologists have been dedicating to study OT protocol. An efficient two-message OT protocol based on decisional Diffie-Hellman(DDH) assumption had been constructed by Naor and Pinkas[5][6] and Aiello, Ishai and Reingold[7]

* This work is supported by the National Natural Science Foundation of China (No.61070250), and Beijing Natural Science Foundation (N0.4132066), and the 12th Five-year Cryptography Development Foundation of China (No.MMJJ201101026), and Scientific Research and Postgraduate Training Cooperation Project-Scientific Research Base-New Theory of Block Cipher and Obfuscation and their Application Research.

** To whom correspondence should be addressed.

independently. Peikert, Vaikuntanathan and Waters[2] also presented an efficient composable non-adaptive OT protocol for 1-out-of-2 variant under decisional Diffie-Hellman(DDH), whose primary technology is a dual-mode cryptosystem and our protocol was inspired their skill and develop it, the specific method see below.

Obfuscation has been formally proposed by Barak, Goldreich et al.[8] at the first time. Roughly speaking, program obfuscation aims to make a computer program "unintelligible" while preserving its functionality. Barak et.al defined a notion of virtual black box(VBB) obfuscation and proved that this notion is impossible to realize in general, i.e., some functions are VBB unobfuscatable. In order to avoid impossible results, Barak et.al defined a weaker notion, i.e, indistinguishable obfuscation, concretely, for a class of circuits $\mathcal{C}$, for any two equivalent circuits C_0, C_1 from the class, the two distribution of obfuscations $i\mathcal{O}(C_0)$ and $i\mathcal{O}(C_1)$ should be computationally indistinguishable, and point out that if the circuit class $\mathcal{C}$ has efficiently computable canonical forms, then the computation of that canonical form would already be an indistinguishability obfuscator. More recently, there are several great breakthrough about obfuscation[1][9][10][11][12][13][14], the most important one is that Garg et al.[1] presented an efficient construction of $i\mathcal{O}$ for all circuits, basing security in part on assumptions related to multilinear maps, which they called Multilinear Jigsaw Puzzles.

Lately, Garg and Gentry et al.[15] present a two-round secure multi-party computation(MPC) from indistingushability obfuscation which also use a NIZK proof system, a CCA-secure PKE scheme and embedded use of n-party semi-honest MPC protocol. It is well known that oblivious transfer is an block to construct multi-party computation and inspired by this work, we try to use indistinguishability obfuscation to construct a secure oblivious transfer protocol, which don't require the CCA-secure PKE scheme.

Our Contributions. Our goal in this paper is to construct a secure oblivious transfer protocol from indistinguishable obfuscation. We follow Peikert, Vaikuntanathan and Waters's steps and generalized their result to non-adaptive OT protocol for k-out-of-l variant under decisional Diffie-Hellman(DDH). And at the same time, we find there are a little defective in their scheme and we improve it(In the SetupMessy phase and SetupDec phase, the length of t are not equal, adversary may distinguish SetupMessy from SetupDec when performing one of these two operation, then it maybe lead to the leak of some information). But our protocol take obf-branch instead of dual-mode as a block to construct the protocol.

The paper is organized as follows: Section 2 gives preliminaries which contain five parts; Section 3 constructs a obfuscator for the encryption scheme based on DDH; Then section 4 proposes a new oblivious transfer protocol from indistinguishability obfuscation. We give security proof in section 5 and conclude this paper in section 6.

2 Preliminaries

In this section, we present the hardness assumption and the definitions of basic blocks used in the whole paper. We let $\mathbb{N}$ denote the natural numbers. For $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, \ldots, n\}$. We let $ploy(n)$ denote an unspecified function $f(n) = O(n^c)$ for some constant c. The security parameter will be denoted by n throughout the paper. We let $negl(n)$ denote some unspecific function $f(n)$ such that $f = o(n^{-c})$ for every fixed c, saying that such a function is negligible. We say that a probability is overwhelming if it is $1 - negl(n)$.

2.1 Complexity Assumptions

Let *Setup* be an algorithm that takes as input a security parameter 1^n and outputs a group description $\mathbb{G} = (G, p, g)$, where G is a cyclic group of prime order p and g is a generator of G. The specific definition as follow:

Definition 1. *(DDH Assumption) For every PPT machine D, all sufficiently large $n \in \mathbb{N}$, every random generator $g, h \in G$,*

$$\left| Pr\left[\begin{array}{ll} \mathbb{G} = (G, p, g) \leftarrow Setup(1^n); & \\ a \leftarrow \mathbb{Z}_q; h \leftarrow G; & : decision = 1 \\ decision \leftarrow D(p, (g, h, g^a, h^a)). & \end{array}\right] - Pr\left[\begin{array}{ll} \mathbb{G} = (G, p, g) \leftarrow Setup(1^n); & \\ a \leftarrow \mathbb{Z}_q; b \leftarrow \mathbb{Z}_q; h \leftarrow G; & : decision = 1 \\ decision \leftarrow D(p, (g, h, g^a, h^b)). & \end{array}\right] \right| \leqslant \frac{1}{p(n)}$$

where G is a cyclic group of prime order p.

That is, the tuples (g, h, g^a, h^a) and (g, h, g^a, h^b) are computationally indistinguishable.

2.2 Indistinguishability Obfuscator

In this subsection we represent the notion of indistinguishability obfuscation ($i\mathcal{O}$) which is defined in [8] using candidate multilinear maps by Garg et al. lately.

Definition 2. *(Indistinguishability Obfuscator ($i\mathcal{O}$)) A uniform PPT machine $i\mathcal{O}$ is called an indistinguishability obfuscator for a circuit class $\mathcal{C}_n$ if the following conditions are satisfied:*

- *For all security parameters $n \in \mathbb{N}$, for all $C \in \mathcal{C}_n$, for all inputs x, we have that*

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

– *For any (not necessarily uniform) PPT distinguisher D, there exists a negligible function α such that the following holds: For all security parameters $n \in \mathbb{N}$, for all pairs of circuits $C_0, C_1 \in \mathcal{C}_n$, we have that if $C_0(x) = C_1(x)$ for all inputs x, then*

$$\left|Pr[D(i\mathcal{O}(n, C_0)) = 1] - Pr[D(i\mathcal{O}(n, C_1)) = 1]\right| \leqslant \alpha(n)$$

2.3 Candidate Indistinguishability Obfuscation for all circuits

In FOCS2013, Garg et al.[1] gave the first construction for indistinguishability obfuscation for all circuits. Firstly, they described a candidate construction for indistinguishability obfuscation for NC^1 circuits. The security of this construction is based on a new algebraic hardness assumption. The candidate and assumption use a simplified variant of multilinear maps, which we called *Multilinear Jigsaw Puzzles*. Secondly, they showed how to use indistinguishability obfuscation for NC^1 together with Fully Homomorphic Encryption(with decryption in NC^1) to achieve indistinguishability obfuscation for all circuits.

In this subsection we introduce the construction of candidate indistinguishability obfuscation. Their construction makes use of two primitives: Perfectly Sound Non-Interactive Witness Indistinguishability Proofs and Fully Homomorphic Encryption. They also use the fact that any poly-time circuit computation can be verified by a "low=depth" circuit in NC^1. Let ($Setup_{FHE}$, $Encrypt_{FHE}$, $Eval_{FHE}$, $Decrypt_{FHE}$). Furthermore, they assumed the decryption algorithm $Decrypt_{FHE}$ can be realized by a family of circuits in NC^1 and that the system has perfect correctness.

The construction is described by an $Obfuscate$ algorithm and an $Evaluation$ algorithm:

– $Obfuscate(1^n, C \in \mathcal{C}_n)$: The $Setup_{FHE}$ algorithm takes the security parameter n and computes the following.
  1. Generate the keys$(pk^1_{FHE}, sk^1_{FHE}) \leftarrow Setup_{FHE(1^n)}$ and $(pk^2_{FHE}, sk^2_{FHE}) \leftarrow Setup_{FHE(1^n)}$. If we are using a leveled FHE scheme, the number of levels should be set to be the depth of U_n.
  2. Generate ciphertexts $q_1 = Encrypt_{FHE}(pk^1_{FHE}, C)$ and $q_2 = Encrypt_{FHE}(pk^2_{FHE}, C)$. Here we assume that C is encoded in a canonical form as an l bit string for use by the universal circuit $U_n(\cdot, \cdot)$.
  3. Generate an NC^1 obfuscation for the program $P1^{(sk^1_{FHE}, q_1, q_2)}$ as $P = i\mathcal{O}_{NC^1}(P1^{(sk^1_{FHE}, q_1, q_2)})$.($P1$ see Figure 1.)
  4. The obfuscation components are output as: $\sigma = (P, pk^1_{FHE}, pk^2_{FHE}, q_1, q_2)$.
– $Evaluate$ $(\sigma = (P, pk^1_{FHE}, pk^2_{FHE}, q_1, q_2))$: The $Evaluate$ algorithm takes in the obfuscation output σ and program input m and computes the following.
  1. Compute $e_1 = Eval_{FHE}(pk^1_{FHE}, U_n(\cdot, m), q_1)$ and $e_2 = Eval_{FHE}(pk^2_{FHE}, U_n(\cdot, m), q_2)$.
  2. Compute a low depth proof ϕ that e_1 and e_2 were computed correctly.
  3. Run $P(m, e_1, e_2, \phi)$ and output the result.

$P1$

Given input (m, e_1, e_2, ϕ),$P1^{(sk^1_{FHE,q_1,q_2})}$ proceeds as follows:

1. Check if ϕ is a valid low-depth proof for the NP-statement: $e_1{=}Eval_{FHE}(pk^1_{FHE}, U_n(\cdot, m), q_1) \wedge e_2{=}Eval_{FHE}(pk^2_{FHE}, U_n(\cdot, m), q_2)$
2. If the check fails output 0; otherwise, output $Decrypt_{FHE}(e_1, sk^1_{FHE})$.

Fig.1.

$P2$

Given input (m, e_1, e_2, ϕ),$P2^{(sk^2_{FHE,q_1,q_2})}$ proceeds as follows:

1. Check if ϕ is a valid low-depth proof for the NP-statement: $e_1{=}Eval_{FHE}(pk^1_{FHE}, U_n(\cdot, m), q_1) \wedge e_2{=}Eval_{FHE}(pk^2_{FHE}, U_n(\cdot, m), q_2)$
2. If the check fails output 0; otherwise, output $Decrypt_{FHE}(e_2, sk^2_{FHE})$.

Fig.2.

The correctness and security of the candidate indistinguishability obfuscation have been proved in [1].

2.4 Randomization

Let G be an arbitrary multiplicative group of prime order p. For each $x \in \mathbb{Z}_p$, we define $L_G(x) = ((g, g^x), g \in G)$. Let $g, h \in G$ are generators of G. The probabilistic algorithm *Randomize* takes $g, h \in G$ and $g^x, h^x \in G$ as input, then output a pair $(u, v) \in G^2$. The specific progress as follow: Choose $s, t \leftarrow \mathbb{Z}_p$ independently, let $u = g^s h^t$ and $v = (g^x)^s (h^x)^t$, $Randomize(g, h, g^x, h^x)$ output (u, v). Then this algorithm have two properties as below.

- If $(g, g^x), (h, h^x) \in L_G(x)$ for some x, the (u, v) is uniformly random in $L_G(x)$.
- For $x, y \in \mathbb{Z}_p$, if there exist $x \neq y$ such that $(g, g^x) \in L_G(x)$, $(h, h^y) \in L_G(y)$, then (u, v) is uniformly random in G^2.

Firstly, we can write $h = g^l$ for some nonzero $l \in \mathbb{Z}_p$, where g and h are generators of G. Choose $x \in \mathbb{Z}_p$ randomly and suppose (g, g^x) and (h, h^x) belong to $L_G(x)$. Now, $u = g^s h^t = g^{s+lt}$ is uniformly random in G, since g is a generator of G and s is random in $\mathbb{Z}_p$. Furthermore, $v = (g^x)^s (h^x)^t = (g^s h^t)^x = (g^{s+lt})^x$ and thus, $(u, v) \in L_G(x)$.

For $x, y \in \mathbb{Z}_p$ and $x \neq y$, now suppose $(g, g^x) \in L_G(x)$ and $(h, h^y) \in L_G(y)$. Then $u = g^s h^t = g^{s+lt}$ and $v = g^{sx+lyt}$. Because $r(x - y) \neq 0 \in \mathbb{Z}_p$, the expression $s + lt$ and $sx + lyt$ are linearly independent combinations of s and t. Therefore, over the choice of $s, t \in \mathbb{Z}_p$, u and v are uniform and independent in G.

2.5 Dual-Mode Encryption

Our protocol not only based on indistinguishability obfuscation but also dual-mode encryption, which is a probabilistic polynomial time Turing machine Π that contains six algorithms[2]:

- Setup $(1^n, \mu)$: given security parameter n and mode $\mu \in \{0,1\}$, outputs (crs, t). When $\mu = 0$, the $Setup(1^n, 0)$ represents messy mode setup algorithm, denoted by $SetupMessy(1^n)$; When $\mu = 1$, the $Setup(1^n, 1)$ represents decryption mode setup algorithm, denoted by $SetupDec(1^n)$. The crs is a common string for the remaining algorithms, and t is a auxiliary trapdoor value that enables either the **FindMessy** or **TrapKeyGen** algorithm, depending on the selected mode.
- $KeyGen(\sigma, crs)$: given a branch value $\sigma \in \{0,1\}$, outputs (pk, sk) where pk is a public key and sk is a corresponding secret key for message encrypted on branch σ.
- $Enc(pk, b, m, crs)$: given a public key pk, a branch value $b \in \{0,1\}$, and a message $m \in \{0,1\}^h$, outputs a ciphertext c encrypted on branch b.
- $Dec(sk, c, crs)$: given a secret key sk and a ciphertext c, outputs a message $m \in \{0,1\}^h$.
- $FindMessy(t, pk, crs)$: given a trapdoor t and some (possibly even malformed) public key pk, outputs a branch value $b \in \{0,1\}$ corresponding to a messy branch of pk.
- $TrapKeyGen(t, crs)$: given a trapdoor t, outputs (pk, sk_0, sk_1), where pk is a public encryption key and sk_0, sk_1 are corresponding secret decryption keys for branches 0 and 1, respectively.

This encryption cryptosystem sets up in one of two modes, call messy mode and decryption mode. The first crucial security property is that no (efficient) adversary can distinguish the two mode given crs. We will focus on these two mode to obfuscate the encryption scheme and generalize it in multi-message and large branch sense. The specific progress are discussed in later sections.

Definition 3. *(Dual-Mode Encryption).A dual-mode cryptosystem is a tuple of algorithms described above that satisfy the following properties:*

1. ***Completeness for decryptable branch:*** *For every $\mu \in \{0,1\}$, every $(crs, t) \leftarrow Setup(1^n, \mu)$, every $\sigma \in \{0,1\}$, every $(pk, sk) \leftarrow EKG(\sigma)$, and every $m \in \{0,1\}^l$, decryption is correct on branch σ,i.e.,$Dec(sk, Enc(pk, \sigma, m)) = m$. It also suffices for decryption to be correct with overwhelming probability over the randomness of the entire experiment.*
2. ***Indistinguishability of modes:*** *The first output of algorithm SetupMessy and SetupDec are computationally indistinguishable, i.e., $SetupMessy(1^n) \overset{c}{\approx} SetupDec(1^n)$.*
3. ***(Messy mode) Trapdoor identification of a messy branch:*** *For every $(crs, t) \leftarrow SetupMessy(1^n)$ and every (possibly malformed) $pk \leftarrow EKG(\sigma)$, $FindMessy(t, pk)$ outputs a branch value $b \in \{0,1\}$ such that $Enc(pk, b, \cdot)$ is messy. Namely, for every m_0,$m_1 \in \{0,1\}^l$, $Enc(pk, b, m_0) \overset{s}{\approx} Enc(pk, b, m_1)$.*

4. ***(Decryption mode)Trapdoor generation of keys decryptable on both branches:*** *For every* $(crs, t) \leftarrow SetupDec(1^n)$*,* $TrapKeyGen(t)$ *outputs* (pk, sk_0, sk_1) *such that for every* $\sigma \in \{0,1\}$,$(pk, sk_\sigma) \overset{s}{\approx} KeyGen(\sigma)$.

3 Construct the Obfuscator for the encryption scheme based on DDH

3.1 The encryption scheme based on DDH assumption

A probabilistic public key cryptosystem PKE is a probabilistic polynomial time Turing machine Π that contains three algorithm:

(1)EKG: on inputs p generates a pair of pubic-secret key (pk, sk) and outputs the description of two algorithms, E and D such that

(2)E is a probabilistic encryption algorithm: for some constants p , public key pk and a plaintext m, returns the ciphertext c, let G be the message space defined by (p, pk).

(3)D is a deterministic decryption algorithm: for some constants p, secret key sk and ciphertext c, returns the plaintext m. The security of a public-key encryption scheme is indistinguishability of Encryptions against CPAs. The details as follow.

Definition 4. *(Indistinguishability of Encryptions against CPAs) A* PKE *scheme* (EKG, E, D) *satisfies the indistinguishability if the following condition holds: For every PPT machine pair* A_1,A_2*(adversary),every polynomial* $p(\cdot)$*, all sufficiently large* $n \in \mathbb{N}$*,*

$$2 \cdot Pr\left[\begin{array}{l} p \leftarrow Setup(1^n); (pk, sk) \leftarrow EKG(p); \\ (m_1, m_2) \leftarrow A_1(p, pk); b \leftarrow \{0,1\}; c \leftarrow E(pk, m_b); \\ d \leftarrow A_2(p, pk, (m_1, m_2), c); \\ b = d. \end{array}\right] - 1 \leqslant \frac{1}{p(n)}$$

where we assume that A_1 *produces a valid message pair* m_0 *and* $m_0 \in G$.

According to the standard definition above, we have the DDH-based encryption scheme as follows.

- $EKG(1^n)$.
  1. Select $\mathbb{G} = (G, p, g) \leftarrow \mathcal{G}(1^n)$, where G is the message space of the scheme.
  2. Choose a uniformly random element h and exponents $r \leftarrow \mathbb{Z}_p$ in G.
  3. Let $pk = (g, h, g^r, h^r)$ and $sk = r$.
  4. Output (pk, sk).
- $Enc(pk, m)$.
  1. Parse pk as (g, h, g^r, h^r).
  2. Let $(u, v) \leftarrow Randomize(g, h, g^r, h^r)$.
  3. Output the ciphertext $c = (u, v \cdot m)$.
- $Dec(sk, c)$.

1. Parse c as (e_0, e_1).
2. Output $m = e_1/e_0^r$.

Obviously, this encryption scheme satisfy the completeness property. More precisely, $Dec(sk, c) = Dec(sk, Enc(pk, m)) = e_1/e_0^r = (g^r)^s(h^r)^t \cdot m/(g^s \cdot h^t)^r = m$ always sets up.

Theorem 1. *Under DDH assumption, the encryption scheme satisfies the indistinguishability.*

Proof. In order to prove the theorem, we show that if there exist an adversary A_1 can distinguishing the encryptions c_0, c_1 of two message m_0 and m_1 under $pk = (g, h, g^r, h^r)$ with non-negligible probability, then we construct an adversary A_1' that will break the DDH assumption with non-negligible probability as well. Following the encryption scheme, we have

$$c_0 = (u_0, v_0 \cdot m_0) = (g^{s_0}h^{t_0}, (g^r)^{s_0}(h^r)^{t_0} \cdot m)$$

$$c_1 = (u_1, v_1 \cdot m_1) = (g^{s_1}h^{t_1}, (g^r)^{s_1}(h^r)^{t_1} \cdot m)$$

Because of $g^{s_0}h^{t_0}$ and $(g^r)^{s_0}(h^r)^{t_0}$ have the same form $(g^a)^{s_0}(h^a)^{t_0}(a = 1$ or $a = r)$.Now let $w_0 = (g^a)^{s_0}(h^a)^{t_0}$ and $w_1 = (g^a)^{s_1}(h^a)^{t_1}$, and A_1 distinguishing c_0 from c_1, we only need to consider A_1 distinguishing w_0 from w_1. As known above, g, h are generators of G, then there is some $l \in \mathbb{Z}_p$ such that $h = g^l$. Now, we have $w_0 = g^{as_0+lat_0}$ and $w_1 = g^{as_1+lat_1}$. Thus A_1 have power to distinguish the two randomness $as_0 + lat_0$ and $as_1 + lat_1$, where s_i, t_i are uniform randomly selected in $\mathbb{Z}_p$ and $i \in \{0, 1\}$.

A_1' works as follows:

- A_1' receives as input a tuple (g, h, g^a, W), where g, h are random generators of the group G and a are random exponents. The goal of A_1' is to determine whether $W = h^a$.
- A_1' picks the random generator g of group G.
- On receiving two message m_0 and m_1 from A_1, A_1' flips a bit b randomly and sends $c_b = (w_0, w_1 \cdot m)$ as the ciphertext of m_b to A_1.
- A_1 replies with a bit b^*. A_1' simply outputs 1 if $b = b^*$ (i.e., guessing that $W = h^a$); otherwise outputs a random bit(i.e., W is a random parameter).

It is easy to see that when W is random, the encryption result c_b is independent of b and hence the success probability of A_1 is exactly $1/2$ in this case. When $W = h^a$, the encryption result c_b has the same distribution as the u_b. According to the assumption, the adversary $A1$ has advantage at least ϵ . That is, A_1' succeeds in determining whether $W = g^a$ also with non-negligible advantage, A_1' breaks the DDH assumption, then the theorem established.

3.2 The obfuscator for DDH-based encryption scheme

In this section, we want to obfuscate the DDH-based encryption scheme above. Inspired by the dual-model cryptosystem which was proposed by Peikert et.al

in [2], we try to combine it with indistinguishability obfuscation definition to get a useful encryption scheme which we called two-branch obfuscator. Let the circuit C_n to compute the DDH-based encryption, the specific progress of the obfuscator Obf_{C_n} is as follows:

- $Setup(1^n)$: $\mathbb{G} = (G, p, g) \leftarrow \mathcal{G}(1^n)$, where G is the message space of the system.
  - 1-obf-branch: Randomly choose generators $g_0 \leftarrow G$ and $g_1 \leftarrow G$. And Randomly choose $x_0, x_1 \leftarrow \mathbb{Z}_p$ as nonzero exponents where $x_0 \neq x_1$. Let $\delta_1 = Obfuscate(1^\lambda, C_1)$ and $\delta_2 = Obfuscate(1^\lambda, C_2)$ (for detail about C_1 and C_2 see Figure 3 and Figure 4). Let $crs = (g_0, \delta_1, g_1, \delta_2)$ and $t = (x_0, x_1)$. Output (crs, t).
  - 2-obf-branch: Randomly choose generators $g_0 \leftarrow G$. And Randomly choose $x, y_0, y_1 \leftarrow \mathbb{Z}_p$ as nonzero exponents where $y_0 \neq y_1$. Let $g_1 = g_0^{y_0}$ and $\delta_1 = Obfuscate(1^\lambda, C_3)$, $\delta_2 = Obfuscate(1^\lambda, C_4)$(for detail about C_3 and C_4 see Figure 5 and Figure 6).Let $crs = (g_0, \delta_3, g_1, \delta_4)$ and $t = (y_0, y_1)$. Output (crs, t).
- $EKG(\sigma, crs)$: Call the $Evaluate(\delta, crs)$ algorithm.Let $h_0 = Evaluate(\delta_1, g_0)$ and $h_1 = Evaluate(\delta_2, g_1)$. Choose uniformly random elements $r \leftarrow \mathbb{Z}_p$. Let $g = g_\sigma^r$ and $h = h_\sigma^r$. Let $pk = (g, h)$. Let $sk = (a, r)$. Output (pk, sk).
- $Enc(pk, b, m, crs)$: Parse pk as (g, h). Let $pk_b = (g_b, h_b, g, h)$. Let $(u, v) \leftarrow Randomize(g, h, g^r, h^r)$. Output the ciphertext $c = (u, v \cdot m)$ as the encryption of m on encrypt-branch b.
- $Dec(sk, c, crs)$: Parse c as (e_0, e_1). Output $m = e_1/e_0^r$.
- $FindMessy(t, pk, crs)$: Parse the 1-obf-branch trapdoor t as (x_0, x_1) where $x_0 \neq x_1$. Parse the public key pk as (g, h). If $h \neq g^{x_0}$, then output $b = 0$ as a (candidate) messy encrypt-branch. Otherwise, we have $h = g^{x_0} \neq g^{x_1}$ because $x_0 \neq x_1$, so output $b = 1$ as a (candidate) messy encrypt-branch.
- $TrapKeyGen(t, crs)$: Parse the 2-obf-branch trapdoor t as a nonzero $y \in \mathbb{Z}_p$. Pick a random $r \leftarrow \mathbb{Z}_p$ and compute $pk = (g_0^r, h_0^r)$ and output $(pk, r, r/y)$.

CIRCUIT C_1

Circuit C_1 proceeds as follows:
INPUT: g_0 selected randomly in 1-obf-branch.
CONSTANTS: x_0 selected randomly in 1-obf-branch.

1. Let $\eta = g_0^{x_0}$
2. Output η.

Fig.3.

CIRCUIT C_2

Circuit C_2 proceeds as follows:
INPUT: g_1 selected randomly in 1-obf-branch.
CONSTANTS: x_1 selected randomly in 1-obf-branch.

1. Let $\eta = g_1^{x_1}$
2. Output η.

Fig.4.

CIRCUIT C_3

Circuit C_3 proceeds as follows:
INPUT: g_0 selected randomly in 1-obf-branch.
CONSTANTS: x selected randomly in 1-obf-branch.

1. Let $\eta = g_0^{x}$
2. Output η.

Fig.5.

CIRCUIT C_4

Circuit C_4 proceeds as follows:
INPUT: g_1 made by g_0 in 1-obf-branch.
CONSTANTS: x selected randomly in 1-obf-branch.

1. Let $\eta = g_1^{x}$
2. Output η.

Fig.6.

Obviously, the string crs of the two obf-branches are computational indistinguishable and the functionality is remained. So we omit the proofs.

3.3 Multi-branch Obfuscation

In this section, we use a Multielement-randomization based on the hardness of Decisional Diffie-Hellman to construct an encryption scheme called DDHEncryption. We note that the Multielement-randomization is evolved by Naor's[5] scheme, the algorithm writes MR for short.

Lemma 1. *(**Multielement-randomization**) Let G be an arbitrary multiplicative group of prime order p. For $(x_1, x_2, \ldots, x_l) \in \mathbb{Z}_p$, we define $L_G(x_1, x_2, \ldots, x_l) = ((g, g^{x_1}, \ldots, g^{x_l}), g \in G)$. The probabilistic algorithm **MR** takes generators g, h and $(g_1, g_2, \ldots, g_l), (h_1, h_2, \ldots, h_l) \in G^l$ as input, output a pair $(u, v) \in G^{n \times n}$, where $u = (u_1, u_2, \ldots, u_l)$ and $v = (v_1, v_2, \ldots, v_l)$.*

– *If for some* $(x_1, x_2, \ldots, x_l)$*, exists* $(g, g_1, g_2, \ldots, g_l) \in L_G(x_1, x_2, \ldots, x_l)$*, and* $(h, h_1, h_2, \ldots, h_l) \in L_G(x_1, x_2, \ldots, x_l)$*, the* (u, v) *is uniformly random in* $L_G(x_1, x_2, \ldots, x_l)$*.*
– *If* $(g, g_1, g_2, \ldots, g_l) \in L_G(x_1, x_2, \ldots, x_l)$*,* $(h, h_1, h_2, \ldots, h_l) \in L_G(y_1, y_2, \ldots, y_l)$ *for* $x_i \neq y_j$*, where* $i, j \in [1, l]$*, then* (u, v) *is uniformly random in* G^{n^2}*.*

Proof. We define the $MR(g, h, g_1, g_2, \ldots, g_l, h_1, h_2, \ldots, h_l)$ proceeds as follow. Select random number $s_i, t_i \in \mathbb{Z}_p$ independently, and let $u_i = g^{s_i} h^{t_i}$, $v_i = {g_i}^{s_i} {h_i}^{t_i}$, where $i \in [1, l]$. Since g and h are generators of G, then for some nonzero $r \in \mathbb{Z}_p$, we have $h = g^r$.

– For some $(x_1, x_2, \ldots, x_l)$, Let $(g, g_1, g_2, \ldots, g_l) \in L_G(x_1, x_2, \ldots, x_l)$, and let $(h, h_1, h_2, \ldots, h_l) \in L_G(x_1, x_2, \ldots, x_l)$, then we have $u_i = g^{s_i} h^{t_i} = g^{s_i + r t_i}$ is uniformly random in G as for the generator g, s_i is random in $\mathbb{Z}_p$, where $i \in [1, l]$. At the same time, $v_i = g_i^{s_i} h_i^{t_i} = g^{s_i x_i} h^{t_i x_i} = (g^{s_i} h^{t_i})^{x_i} = u_i^{x_i}$, where $i \in [1, l]$, $(u, v) \in L_G(x_1, x_2, \ldots, x_l)$.
– And for $x_i \neq y_j$, then we let $(g, g_1, g_2, \ldots, g_l) \in L_G(x_1, x_2, \ldots, x_l)$, and let $(h, h_1, h_2, \ldots, h_l) \in L_G(y_1, y_2, \ldots, y_l)$. Then we have $u_i = g^{s_i} h^{t_i} = g^{s_i + r t_i}$ and $v_i = (g_i)^{s_i} (h_i)^{t^i} = g^{x_i s_i} h^{y_i t_i} = g^{x_i s_i + y_i t_i r}$. Since $r(x_i - y_i) \neq 0 \in \mathbb{Z}$, the expressions $s_i + r t_i$ and $x_i s_i + r y_i t_i$ are linearly independent combinations of s_i and t_i. Therefore, over the choice $s_i t_i \in \mathbb{Z}$, where $i \in [1, l]$, u and v are uniform and independent in G.

Thus, the lemma is established. Then we adjust it to be a multi-encryption cryptosystem.

(**Multi-encryption Scheme**) Here we propose a new defition called **Multi-branch Obfuscation** informally, where the branch refers to the encrypt-branch σ in the above cryptosystem. For $i \in [1, l]$:

1. DDHKeyGen(1^n). Select $\mathbb{G} = (G, p, g) \leftarrow \mathcal{G}(1^n)$. The message space of the scheme is G. Choose a uniformly random element h and exponents $(x_1, x_2, \ldots, x_n) \leftarrow \mathbb{Z}_p$ in G. Let $pk_i = (g, h, g^{x_i}, h^{x_i})$ and $sk_i = x_i$, then output (pk_i, sk_i).

2. DDHEnc(pk_i, m_j). Parse pk_i as (g, h, g_i, h_i). Let $(u_i, v_i) \leftarrow Randomize(g, h, g_i, h_i)$ and output the ciphertext $(u_i, v_i \cdot m_i)$,where j.

3. DDHDec(sk_i, c_i). Parse c_i as $(c_{0,i}, c_{1,i})$ and output $c_{1,i} / c_{0,i}^{sk_i}$.

As above described, every branch is similar each other and can be seen as indistinguishable. So we can call it a simple **Multi-branch Obfuscation** temporarily.

4 The oblivious protocol from indistinguishability obfuscation

4.1 Oblivious Transfer Protocol

Now we will define oblivious transfer protocol formally. Obvious transfer protocol involves a sender Alice with input $x_1, \ldots, x_l$ and a receiver Bob with input

$\sigma_1, \ldots, \sigma_k \in [1, l]$. Bob should learn $x_{\sigma_1}, \ldots, x_{\sigma_k}$ (and nothing else) and Alice should learn nothing. The exact definition of the ideal oblivious transfer functionality, denoted $\mathcal{F}_{OT}^{k \times l}$ which capture k-out-of-l OT, specific progress see Figure 7.

Functionality $\mathcal{F}_{OT}^{k \times l}$

$\mathcal{F}_{OT}^{k \times l}$ proceeds as follow, parameterized with an integer l and interacts with sender Alice and receiver Bob and adversary $\mathcal{S}$:

1. Upon receiving a message (**sid, sender**, $x_1, \ldots, x_l$) from **S**, where each $x_i \in \{0,1\}^m$, record the tuple $(x_1, \ldots, x_l)$.(The length of the strings m is fixed and known to all parties.)
2. Upon receving a message (**sid, reciver**, $\sigma_1, \ldots, \sigma_k$) from R, where $\sigma_1, \ldots, \sigma_k \in [1, l]$, check if a (**sid, receiver**,...) message was previously sent. If yes, send (**sid,**x_{σ_1}**,...,**x_{σ_k}) to **R** and (**sid**) to adversary $\mathcal{S}$ and halt. If not, send nothing to **R**(but continue running).

Fig.7. The oblivious transfer functionality $\mathcal{F}_{OT}^{k \times l}$

4.2 Construct the Oblivious Transfer Protocol from indistinguishability obfuscation

Protocol OT^{ob} for Oblivious Transfer

OT^{ob} proceeds as follows, parameterized with the $ob \in \{1, 2\}$ that chosen previously (which indicates the type of crs to be used, i.e. ob=1 means choose 1-obf-branch and ob=2 means 2-obf-branch) and an integer l, it interacts with sender Alice and receiver Bob and adversary $\mathcal{S}$:

SENDER INPUT: (**sid, sender**, $x_1, \ldots, x_l$), where $x_1, \ldots, x_l \in \{0,1\}^n$

RECEIVER INPUT: (**sid, reciver**, $\sigma_1, \ldots, \sigma_k$), where $\sigma_1, \ldots, \sigma_k \in \{0, l\}$

.

When actived with their input, the sender **S** queries $\mathcal{F}_{CRS}$ with (**sid**, **S**,**R**) and gets back (**sid**,crs). The receiver **R** then queries $\mathcal{F}_{CRS}$ with (**sid**, **S**,**R**) and gets back (**sid**,crs).

1. **R** computes $(pk_i, sk_i) \leftarrow EKG(crs, \sigma_i)$ where $i \in [1, k]$, sends (**sid**, **ssid**, pk_1 ,...,pk_k) to **S** and stores (**sid**, **ssid**, sk_1 ,...,sk_k).
2. **S** gets (**sid**, **ssid**, pk_1 ,...,pk_k) from **R**, computes $y_{i,b} \leftarrow Enc(pk_i, b, x_b)$ for each $b \in [1, l]$ where $i \in [1, k]$,and sends (**sid**,**ssid**,$y_{i,1}$,...,y_{il}) to **R** where $i \in [1, k]$.
3. **R** gets (**sid**,**ssid**,$y_{i,1}$,...,$y_{i,l}$) from **S** and outputs (**sid**,**ssid**,$Dec(sk_i, y_{i,\delta_i})$)where $i \in [1, k]$ and (**sid**,**ssid**,sk_1,...,sk_k)was stroed above.

Fig.8. The oblivious transfer protocol OT^{ob} realizing functionality $\mathcal{F}_{OT}^{k \times l}$

Here we construct a protocol OT^{ob} that realizes the ideal oblivious transfer functionality $\mathcal{F}_{OT}^{k\times l}$. The protocol can actually operate in either obf-branches, which only affects the distribution of the CRS that is used. Our protocol is given in Figure 8 above. And our OT protocol operates in common reference string model. By designing a protocol for the multi-session extension $\mathcal{F}_{OT}^{k\times l}$ of the OT functionality $\mathcal{F}_{OT}$, we reuse the same common reference string for distinct invocations of oblivious transfer whenever possible. According to [2], $\mathcal{F}_{OT}^{k\times l}$ acts as a "wrapper" around any number of independent executions of $\mathcal{F}_{OT}$ and coordinates their interactions with the parties via subsessions (specified by a parameter **ssid**) of a single session (specified by a parameter **sid**).

5 The Security of our Obfuscator of DDH-based Cryptosystem and OT^{ob} protocol

In this section, we attribute the security of our cryptosystem to DDH assumption and the dual-mode cryptosystem[2] and the candidate indistinguishability obfuscator[1] consisting the $Obfuscate(1^n, C)$ and $Evaluate(\sigma, m)$ algorithm.

Lemma 2. ***(Indistinguishability of the candidate obfuscation)*** *It is hard to distinguish an obfuscation of C_0 from an obfuscation of C_1, for any two circuits C_0 and C_1 of the same size that compute the exact same function.*

Proof sketch. In [1], they proved that for all $C_0, C_1 \in \mathcal{C}_\lambda$ there can be no poly-time indistinguishability attacker $\mathcal{A}$ that wins the security game from section 2 in [1] with non-negligible advantage. And they also proved that their $i\mathcal{O}$ for poly-sized circuits is secure in the indistinguishability game. They completed their proof using a sequence of hybrids. The challenger obfuscates C_0 in the first hybrid. They then gradually changed the obfuscation in multiple hybrid steps into an obfuscation of C_1. They showed that each successive hybrid experiment is indistinguishable from the last, thus showing the obfuscator to have indistinguishability security. The proof hybrid stepped themselves primarily weave back and forth in between changing the underlying ciphertexts and the programs that are used in a two-key proof type manner. (details of the proof see [1])

Theorem 2. *No adversary can distinguish which obf-branch is used with overwhelming probability, assuming that DDH is hard for $\mathcal{G}$.*

proof sketch.We give out the proof sketch of theorem above as follows.

- Our construction employs groups for which the DDH problem is believed to be hard. We use the DDH assumption whose version is as follows: for random generators g,$h \in G$ and for distinct but otherwise random a,$b \in \mathbb{Z}_p$, the tuple (g, h, g^a, h^a) and (g, h, g^a, h^b) are computational indistinguishable. This version of the DDH assumption is equivalent to another common form, namely, that $c \neq ab$ with overwhelming probability.

– The first security property of our scheme is the indistinguishability of the two obf-branchs. Obviliously, the outputs of two obf-branches $crs_1 = (g_0, \delta_1, g_1, \delta_2)$, $t_1 = (x_0, x_1)$ and $crs_2 = (g_0^{'}, \delta_3, g_1^{'}, \delta_4)$ and $t_2 = (y_0, y_1)$ have the same form: g_0, g_1, $g_0^{'}$ and $g_1^{'}$ are all generators chosen randomly, δ_1, δ_2, δ_3 and δ_4 are all obfuscation of the equivalent circuits, which implies indistinguishable.
– The pk of the two obf-branches is also indistinguishable. In EKG algorithm, the obfuscation is transformed into h_b and $crs = (g_0, h_0, g_1, h_1)$. In 1-obf-branch, $crs = (g_0, h_0 = g_0^{x_0}, g_1, h_1 = g_1^{x_1})$, where g_0, g_1 are random generators of G and x_0,x_1 are distinct and nonzero in $\mathbb{Z}_p$. Let $a{=}log_{g_0}g_1$, which is nonzero but otherwise uniform in in $\mathbb{Z}_p$. Then $b{=}log_{h_0}(h_1){=}a \cdot x_1/x_0$ is nonzero and distinct from a, but otherwise uniform. Therefore crs os statistically closed to a random DDH non-tuple (g_0, h_0, g_0^a, h_0^b), where $a,b \leftarrow \mathbb{Z}_p$. Since $log_{h_0}(h_1){=}log_{g_0}(g_1) = y$ is nonzero and random in $\mathbb{Z}_p$, crs is statically close to a random DDH tuple. Under the DDH assumption, the indistinguishability of crs results to the indistinguishability of the pk.

Theorem 3. *Let $ob \in \{0, 1\}$. Protocol OT^{ob} securely realizes the functionality $\mathcal{F}_{OT}^{k\times l}$ in the $\mathcal{F}_{CRS}$-hybrid model.*

For ob=1, the sender's security is statical and the receiver's security is computational; for ob=2, the security properties are reversed.

Proof. Given all the properties of a dual-mode cryptosystem (1-obf-branch corresponds to the messy mode and 2-obf-branch corresponds to the decryption mode), the proof is conceptually quite straightforward. There is a direct correspondence between completeness and the case that neither party is corrupted, also between 1-obf-branch and statistical security for the sender, and between 2-obf-branch and statistical security for the receiver. The indistinguishability of two obf-branchs will establishes computational security for the appropriate party in the protocol.

S and **R** are the two parities who take part in the communication, and they run the protocol OT^{obra}. Let $\mathcal{A}$ be a static adversary that interacts with the parties. We now construct an ideal world adversary (simulator) $\mathcal{S}$ interacting with the ideal functionality $\mathcal{F}_{OT}^{k\times l}$, such that no environment $\mathcal{Z}$ can distinguish an interaction with $\mathcal{A}$ in the above protocol from an interaction with $\mathcal{S}$ in the ideal world. Recall that $\mathcal{S}$ interacts with both the environment $\mathcal{Z}$ and the ideal functionality $\mathcal{F}_{OT}^{k\times l}$.

$\mathcal{S}$ starts by invoking a copy of $\mathcal{A}$ and running a simulated interaction of $\mathcal{A}$ with $\mathcal{Z}$ and the players **S** and **R**, $\mathcal{S}$ works as follows:

Simulating the communication with $\mathcal{Z}$: Every input value that $\mathcal{S}$ from $\mathcal{Z}$ is written into the adversary $\mathcal{A}'s$ input tape(as if coming from $\mathcal{A}'s$ environment). Every output value written by $\mathcal{A}$ on its output tape is copied to $\mathcal{S}'s$ own output tape(to be read by the environment $\mathcal{Z}$).

Simulating the case when only the receiver R is corrupted: Regardless of the obf-branch of the protocol, $\mathcal{S}$ does the following. Run the 1-obf-branch setup algorithm, producing (crs, t). When the parties query the ideal functionality $\mathcal{F}_{OT}^{k\times l}$, return (sid, crs) to them. (Note that when $obra$=2-obf-branch, the crs

then returned is identically distributed to the one returned by OT^{obra}, whereas when $obra$=1-obf-branch, the simulated crs has a different distribution from the one returned by OT^{obra} in the protocol).

When $\mathcal{A}$ produces a protocol message $(sid, ssid, pk)$, $\mathcal{S}$ find the messy branch letting $b \leftarrow FindMessy(crs, t, pk)$. $\mathcal{S}$ then sends $(sid, ssid, receiver, 1-b)$ to the ideal functionality $\mathcal{F}_{OT}^{k\times l}$, receives the output $(sid, ssid, x_{1-b})$, and stores it along with the value b.

When the dummy $\mathbf{S}$ is activated for subsession $(sid, ssid)$, $\mathcal{S}$ looks up the corresponding b and x_{1-b}, computes $y_{1-b} \leftarrow Enc(pk, b, 0^l)$ and sends the adversary $\mathcal{A}$ the message $(sid, ssid, y_0, y_1)$ as if it were from $\mathbf{S}$.

Simulating the case when only the send S is corrupted: Regardless of the obf-branch of the protocol, $\mathcal{S}$ does as follows. Run the 2-obf-branch setup algorithm, producing (crs, t). When the parities query the ideal functionality $\mathcal{F}_{OT}^{k\times l}$, return (sid, crs) to them.

When the dummy $\mathbf{R}$ is actived on $(sid, ssid)$, $\mathcal{S}$ computes $(pk, sk_0, sk_1) \leftarrow TrapKeyGen(crs, t)$, sends $(sid, ssid, pk)$ to $\mathcal{A}$ as if from $\mathbf{R}$, and stores $(sid, ssid, pk, sk_0, sk_1)$. When $\mathcal{A}$ replies with a message $(sid, ssid, y_0, y_1)$, $\mathcal{S}$ looks up the corresponding (pk, sk_0, sk_1), computes $x_b \leftarrow Dec(sk_b, y_b)$ for each $b \in 0, 1$ and sends to $\mathcal{F}_{OT}^{k\times l}$ the message $(sid, ssid, sender, x_0, x_1)$.

Simulating the remaining cases: When both parties are corrupted, the simulator $\mathcal{S}$ just runs $\mathcal{A}$ internally(who itself generates the message from both $\mathbf{S}$ and $\mathbf{R}$).

When neither party is corrupted, $\mathcal{S}$ internally runs the honest $\mathbf{R}$ on input $(sid, ssid, \delta = 0)$ and honest $\mathbf{S}$ on input $(sid, ssid, x_0 = 0^l, x_1 = 0^l)$, activating the appropriate algorithm when the corresponding dummy party is activated in the ideal execution, and delivering all messages between its internal $\mathbf{R}$ and $\mathbf{S}$ to $\mathcal{A}$. The proof will be completed using the following claims, which are proved lately:

1.(Claim 1, statistical security for $\mathbf{S}$ in 1-obf-branch) When $\mathcal{A}$ corrupts the receiver $\mathbf{R}$,

$$IDEA_{\mathcal{F}_{OT}^{k\times l},\mathcal{S},\mathcal{Z}} \overset{s}{\approx} EXEC_{OT^1,\mathcal{A},\mathcal{Z}}$$

2.(Claim 2, statistical security for $\mathbf{R}$ in 2-obf-branch) When $\mathcal{A}$ corrupts the receiver $\mathbf{S}$,

$$IDEA_{\mathcal{F}_{OT}^{k\times l},\mathcal{S},\mathcal{Z}} \overset{s}{\approx} EXEC_{OT^2,\mathcal{A},\mathcal{Z}}$$

3.(Claim 3, parameter switching.) For any protocol π^{ob} in the $\mathcal{F}_{CRS}$-hybrid model, any adversary $\mathcal{A}$ and any environment $\mathcal{Z}$,

$$EXEC_{\pi^1,\mathcal{A},\mathcal{Z}} \overset{c}{\approx} EXEC_{\pi^2,\mathcal{A},\mathcal{Z}}$$

We now complete the proof as follows. Consider the protocol OT^1. When $\mathcal{A}$ corrupts $\mathbf{R}$, by item 1 above we have statistical security for $\mathbf{S}$(whether or nor $\mathbf{S}$ is corrupted). When When $\mathcal{A}$ corrupts $\mathbf{S}$, by item 2 and 3 we have

$$IDEA_{\mathcal{F}_{OT}^{k\times l},\mathcal{S},\mathcal{Z}} \overset{s}{\approx} EXEC_{OT^2,\mathcal{A},\mathcal{Z}} \overset{c}{\approx} EXEC_{OT^1,\mathcal{A},\mathcal{Z}}$$

which implies computational security for $\mathbf{R}$.

It remains to show computation security when neither the sender nor the receive is corrupted. Let $EXEC_{OT^1,\mathcal{A},\mathcal{Z}}(x_0, x_1, b)$ denote the output of an envi-

ronment in the protocol OT^1 that sets the inputs of the sender $\mathbf{S}$ to be the bit b. The following sequence of hybrids establishes what we want.

$$EXEC_{OT^1,\mathcal{A},\mathcal{Z}}(x_0,x_1,b)\overset{s}{\approx}EXEC_{OT^1,\mathcal{A},\mathcal{Z}}(0^l,x_1,1)\overset{c}{\approx}$$
$$EXEC_{OT^1,\mathcal{A},\mathcal{Z}}(0^l,x_1,0)\overset{s}{\approx}EXEC_{OT^1,\mathcal{A},\mathcal{Z}}(0^l,0^l,0)$$

The first two and the last two experiments are statically indistinguishable because of the messy property of encryption, and the second and third experiments are computational indistinguishable because of the computational hiding of the receiver's selection bit.The first experiment corresponds to the real world execution, whereas the last experiment is what the simulator runs. Furthermore, by the completeness of the dual-mode cryptosystem, the first experiment is statistically indistinguishable from the ideal world execution with input (x_0,x_1,b).

The proof of security for protocol OT^2 follows symmetrically, and we are done.

It remains to prove the three claims made in the proof above.

Claim 1. If the adversary $\mathcal{A}$ corrupts the receiver $\mathbf{R}$ in an execution of OT^1, then we have

$$IDEA_{\mathcal{F}_{OT}^{k\times l},\mathcal{S},\mathcal{Z}}\overset{s}{\approx}EXEC_{OT^1,\mathcal{A},\mathcal{Z}}$$

Proof. The real world execution can be seen as a game that proceeds as follows, interacting with the environment $\mathcal{Z}(z)$: first,produce crs. Then the environment arbitrarily schedules some number of subsessions, where in each subsession, $\mathcal{Z}$ chooses an arbitrary pk and arbitrary inputs (x_0,x_1) for the honest sender $\mathbf{S}$, who sends $y_b\leftarrow Enc(crs,pk,b,x_b)$ for each $b\in\{0,1\}$ to $\mathcal{Z}$.

The ideal world excution proceeds similarly; however,first produce (crs,t)(but only crs is visible to $\mathcal{Z}$). Then the environment arbitrarily schedules subsessions, where in each subsession $\mathcal{Z}$ produces an arbitrary pk and arbitrary input (x_0,x_1) for the dummmy $\mathbf{S}$. The simulator $\mathcal{S}$ runs $b\leftarrow FindMessy(t,pk)$ and learns x_{1-b} from the functionality $\mathcal{F}_{OT}^{k\times l}$. It then sends $y_b\leftarrow Enc(crs,pk,b,0^l)$ and $y_{1-b}\leftarrow Enc(crs,pk,1-b,x_{1-b})$ to $\mathcal{Z}$.

The only difference between the two games is therefore in y_b in each subsession. But by trapdoor identification of a messy branch, we have in the ideal game that $Enc(crs,pk,b,0^l)\overset{s}{\approx}Enc(crs,pk,b,x_b)$. Therefore the two games are statistically indistinguishable.

Claim 2. If the adversary $\mathcal{A}$ corrupts the receiver $\mathbf{S}$ in an execution of OT^2, then we have

$$IDEA_{\mathcal{F}_{OT}^{k\times l},\mathcal{S},\mathcal{Z}}\overset{s}{\approx}EXEC_{OT^2,\mathcal{A},\mathcal{Z}}$$

The proof method is as above, so we omit this proof.

Claim 3. For any protocol π^{ob} in the $\mathcal{F}_{CRS}$-hybrid model, any adversary $\mathcal{A}$ and any environment $\mathcal{Z}$,

$$EXEC_{\pi^1,\mathcal{A},\mathcal{Z}}\overset{c}{\approx}EXEC_{\pi^2,\mathcal{A},\mathcal{Z}}$$

Proof. By the indistinguishability of the two obf-branch, the crs are computational indistinguishable. Environment $\mathcal{Z}$ running protocol π^{ob} can be seen as an efficient algorithm that receives a polynomial number of samples from either 1-obf-branch or 2-obf-branch. By a standard hybrid argument, the two executions are indistinguishable.

6 Conclusion

A new oblivious transform protocol from indistinguishability obfuscation has been proposed in this paper under DDH assumption and based on the the dual-mode cryptosystem. Following Chris Peikert, Vinod Vaikuntanathan and Brent Waters's steps, we presents our new DDH cryptosystem using the candidate indistinguishability obfuscation of [1] and dual-mode model. Our scheme can protect the privacy of senders and receivers realizing the oblivious transform functionality, and we then give proof of its security. Furthermore, we will continue to focusing the research and application of obfuscation.

References

1. Chris Peikert, Vinod Vaikuntanathan, Brent Waters. A framework for efficient and composable oblivious transfer. In CRYPTO, Advances in Cryptology-CRYPTO 2008, 28st Annual International Cryptology Conference, Santa Barbara, California,USA, August 17-21, 2008. Proceedings,pages 554-571, 2008.
2. Samjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In FOCS, 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS) 2013, 27-29 October 2013, Berkeley, California, USA. Proceedings, pages 40-49, 2013.
3. Michael O. Rabin. How to exchange secrets by oblivious transfer. Technical report, Harvard University, 1981.
4. Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. In CRYPTO, Advances in Cryptology Conference, anta Barbara, California, USA, pages 205-210, 1985.
5. Moni Noar and Benny Pinkas. Oblivious transfer with adaptive queries. In CRYPTO, Advances in Cryptology Conference, Santa Barbara, California, USA, December 16, 2001. Proceedings, pages 573-590, 1999.
6. Moni Noar and Benny Pinkas. Efficient oblivious transfer protocols. In SODA, pages 448-457, 2001.
7. William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In EUROCRYPT, Advances in Cryptoloty-EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001. Proceedings, pages 119-135, 2001.
8. Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P.Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In CRYPTO, Advances in Cryptology-CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California,USA, August 19-21, 2008. Proceedings,pages 1-18, 2001.
9. Nir Bitansky, Ran Canetti, Henry Cohn, Shafi Goldwasser, Yael Tauman Kalai, Omer Paneth, and Alon Rosen. The impossibility of obfuscation with auxiliary input or a universal simulator. In CRYPTO, Advances in Cryptology-CRYPTO 2014, 34th Annual International Cryptology Conference, Santa Barbara, California,USA, August 17-21, 2014. Proceedings,pages 71-89, 2014.
10. Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In EUROCRYPT, Advances in

Cryptology-EUROCRYPT 2014, 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Preceedings, pages 221-238, 2014.
11. Zvik Brakerski and Guy N.Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In TCC, 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings, pages 1-25, 2014.
12. Ilan Komargodski, Tal Moran, Moni Noar, Rafael Pass, Alon Rosen, and Eylon Yogev. One-way functions and (im)perfect obfuscation. In 55th IEEE Annual Symposium on Foundations of Computer Sciencem, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014, pages 374-383. IEEE Computer Society, 2014.
13. Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In CRYPTO, Advances in Cryptology-CRYPTO 2014, 34th Annual International Cryptology Conference, Santa Barbara, California,USA, August 17-21, 2014. Proceedings,pages 500-517, 2014.
14. Benny Applebaum and Zvika Brakerski. Obfuscating circuits via composite-order graded encoding. In TCC 2015, Theory of Cryptography, 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015. Proceedings, pages 528-556, 2015.
15. Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In TCC 2014, Theory of Cryptography, 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA. Feburary 24-26, 2014. Proceedings, pages 79-94, 2014.