Public-Key Encryption from Average Hard NP Language

Hongda Li[1], Peifang Ni[2,3], and Yao Zan[1]

1. State Key Laboratory of Information Security, Institute of Information Engineering, CAS
2. Institute of Software, Chinese Academy of Sciences
3. Zhongguancun Laboratory, Beijing
lihongda@iie.ac.cn, peifang2020@iscas.ac.cn,zanyao@iie.ac.cn

Abstract. The question of whether public-key encryption (PKE) can be constructed from the assumption that one-way functions (OWF) exist remains a central open problem. In this paper we give two constructions of bit PKE scheme derived from any NP language L, along with a polynomial-time instance-witness sampling algorithm. Furthermore, we prove that if L is average hard NP language, the the presented schemes is CPA secure. Our results give a positive answer to this longstanding problem, as the existence of OWF implies the existence of average hard NP language with a polynomial-time instance-witness sampling algorithm.
Additionally, we obtain a witness encryption (WE) scheme for NP language based on the presented PKE scheme. This result highlights that WE scheme can also be established based on the existence of OWF.

Keywords: public-key encryption, one-way function, average hard NP language, witness encryption.

1 Introduction

Public-key encryption (PKE) stands as a crucial primitive in modern cryptography, first introduced by Diffie and Hellman [10] in 1976. Following the work of [10], numerous works have explored different approaches to constructing PKE based on various assumptions (such as integer factoring problem, discrete logarithm problem, LWE, et.al) [3]. In recent, Berman et al. [4] present a PKE construction under a general complexity assumption that average hard NP language (with a polynomial-time instance-witness sampling algorithm) has a laconic SZK argument system. As discussed in [4], the laconism requirement for SZK arguments may make the assumption stronger than the existence of OWF. This observation is supported by the work of Nguyen et al. [20], demonstrating that the existence of OWF implies every language in NP has a SZK augment. Given that currently known constructions of PKE rely on specialized assumptions believed to be stronger than the existence of OWF, which is both necessary and sufficient for private key encryption [12, 16], it is natural to ask what assumption is minimum for realizing PKE and whether PKE can be constructed from OWF.

The earliest exploration of this issue came from R. Impagliazzo and S. Rudich [18]. Unfortunately, they established the impossibility of constructing PKE from OWFs in fully black-box manner (black-box construction and black-box reduction)[22]. The result of [18] indicates that any PKE construction relying on OWF must rely on non-black-box techniques. In fact, non-black-box techniques are widely recognized as more powerful than black-box technique and

can be used to break through the impossibility of black-box technique [2, 7, 19, 21]. However, we do not currently know how to construct public-key encryption schemes based on OWF, and only a few works have demonstrated the non-black-box separation between PKE and OWF. Brakerski et al. in [6] proved limitations of a class of non-black-box techniques (specifically those related to zero-knowledge proofs relative to some oracles) for constructing PKE from OWF. Dachman-Soled [9] considered a special non-black-box technique (black-box construction and non-black-box reduction) and showed the impossibility of obtaining PKE from OWF. S. Garge and M. Hajiabadi in their recent work [14] studied the possibility of constructing PKE using Yao's garbling techniques (a non-black-box technique based on OWF) and proved that garbling circuit that has OWF gates is insufficient for PKE. However, these negative answers presented in [6, 9, 14] do not rule out the possibility of basing PKE on the existence of OWF by other special non-black-box techniques. As a result, the question of whether PKE can be constructed based on OWF remains one of the central open problems in the field of cryptography.

In work [17], Impagliazzo described five possible worlds we could live in, and used the word "Minicrypto" for the world in which OWFs exist but public-key cryptography does not and word "Cryptomania" for the world in which public-key cryptography is possible. Clearly, the open problem (whether PKE can be constructed based on OWF) is in fact whether "Cryptomania"="Minicrypto". The non-black-box separations of [6, 9, 14] may be viewed as an indication that "Cryptomania" and "Minicrypto" are different. However, it remains uncertain whether there an inherent gap exists between "Cryptomania" and "Minicrypto".

This paper centers its attention on exploring the possibility of constructing PKE based on OWF. Considering that the existence of one bit encryption is sufficient to construct PKE schemes for messages of arbitrary length, we only consider bit encryption scheme. By giving two non-black-box constructions of PKE from average hard NP language with an efficient sampling algorithm $\mathcal{S}$, we give a positive answer to the aforegoing open question.

1.1 Our Contributions

In this paper, we present two constructions of PKE scheme based on an NP language L with a polynomial-time sampling algorithm that output instance-witness pair. We prove that, if L is average hard, the presented PKE schemes are CPA secure. It is well known that the existence of average hard NP language with a instance-witness sampling algorithm is equivalent to the existence of OWF. Therefore our results confirm in fact that PKE can be based on OWF, that is, "Cryptomania" and "Minicrypto" are actually the same world.

We now sketch our schemes. Assume that L is average hard NP language with a sampling algorithm $\mathcal{S}$ and NP relation R_L. Let $(x, w) \leftarrow \mathcal{S}(1^\lambda)$, where λ is security parameter, $|x| = \ell, |w| = p$. In our PKE schemes, instance $x \in L$ and security parameter λ are used as public key pk, and the witness w of x (satisfying $R_L(x, w) = 1$) is used as private key sk. When encrypting b ($b = 0, 1$), the encryption algorithm Enc (taking as inputs pk and b) generates a ciphertext $\widehat{C}_b$ which is a garbled circuit (called as constrained-partial-input garbled circuit). When decrypting a ciphertext $\widehat{C}_b$, the decryption algorithm Dec (taking as inputs sk and $\widehat{C}_b$) compute $\widehat{C}_b$ on inputting sk, $b' = \widehat{C}_b(sk)$, and outputs a decrypted message b'. For the correctness and security of the scheme, obviously, $\widehat{C}_b$ must satisfies following conditions:

1) When $R_L(x, w) = 1$, $\widehat{C}_b(w)$ reveals b, that is, $\widehat{C}_b(w) = b$;

2) Without w satisfying $R_L(x,w)=1$, it is hard to obtain b from $\widehat{C}_b$, that is, $\widehat{C}_0$ and $\widehat{C}_1$ are computationally indistinguishable.

In order to obtain $\widehat{C}_b$, we first construct a random circuit $C(u,v):\{0,1\}^\ell\times\{0,1\}^p\to\{0,1\}$ by randomizing NP relation $R_L(x,w)$, and then garble C to generate $\widehat{C}_b$.

Here, We emphasize that we use a new garbling circuit scheme which is essentially Yao's garbled circuit scheme but slightly changed. $\widehat{C}_b$ contains two encoding tables for u and v respectively, all garbled logical gates, and a decoding table. Note that we need to constrain u in $\widehat{C}_b$ taking a random value $\widetilde{u}$ (therefore, $\widehat{C}_b$ is called as constrained-partial-input garbled circuit), so the encoding table for u only contains the encoding of this random value $\widetilde{u}$. In following, we use $\widehat{C}_b(v)=\widehat{C}_b(\widetilde{u},v)$ to mean to compute $\widehat{C}_b$ on inputting $(\widetilde{u},v)$. In order to meet the above two conditions, our first scheme makes $\widehat{C}_0$ and $\widehat{C}_1$ have the same encoding table for v but different values $\widetilde{u}$, while our second scheme makes $\widehat{C}_0$ and $\widehat{C}_1$ have different encoding tables for v but the same value $\widetilde{u}$.

Yao's garbled circuit is used for secure two-party computation, in where both parties wish to carry out a joint computation of a shared function without disclosing their private input. The scenario we consider in the paper is different from secure two-party computation. Specifically, in our PKE scheme the function and the circuit C to compute it are both randomly generated during encryption and so are not public. Moreover, the garbled circuit $\widehat{C}_b$ (ciphertext) is only used for decryption and does not need to be verified at all, so C can contain non-standard logic gates (other than AND, OR, NOT, NAND and NOR). This also makes it easier to hide the information of C through Yao's garbled circuit scheme.

In addition, notice that Yao's garbled circuit scheme requires a private key encryption scheme, while our new scheme does not. $\widehat{C}_b$ contains the encoding table of v, so any one who obtains $\widehat{C}_b$ can compute $\widehat{C}_b(v)$ for any v. This results in the fact that private key encryption scheme is no longer needed when garbling C. In other words, each garbled logic gates in $\widehat{C}_b$ is in fact its truth table represented by the encoding corresponding to the input and output. In particular, the truth tables may be incomplete since u is constrained to be $\widetilde{u}$.

We highlight that the first construction of bit PKE scheme is a witness encryption (WE). This result shows that WE scheme can also be based on the existence of OWF. Moreover, S. Garge et al. in [13] presented ways to obtain non-black-box constructions of PKE and identity-based encryption (IBE) and attribute-based encryption (ABE) from WE and OWF. Therefore, combining the second construction with the results of [13], we can further obtain non-black-box constructions of PKE, IBE and ABE from OWF.

1.2 About Witness Encryption

Witness encryption (WE), proposed by Garge et al. in 2013 [13], is a encryption scheme for some NP language. In WE defined for L with NP relation R_L, the encryption algorithm take as inputs an instance x and a message to generates a ciphertext. A ciphertext relative to x can be decrypted correctly using a witness w for $x\in L$ (satisfying $R_L(x,w)=1$) and hides the message when $x\notin L$. WE is a powerful cryptographic primitive and can be used to construct many other cryptographic schemes. Garge et al. in [13] show non-black-box constructions of PKE, IBE and ABE from WE. Therefore, if it is possible to construct WE scheme from OWF, then the results of [13] would show a possible way to construct PKE from OWF. However, all known constructions of WE rely on stronger assumptions, and the works of [13, 15, 8]

construct WE scheme from multilinear maps, and the work of [11] constructs WE scheme from indistinguishability obfuscation (iO) [1].

1.3 Organization of the Paper

In section 2 we present notions that we need. Specifically, we detail a new garbled circuit scheme (called as constrained-partial-input garbled circuit) which is essentially Yao's garbled circuit scheme but slightly changed. In section 3 we formally describe our PKE scheme and proofs. Section 4 is an extension of section 3 and obtains a WE scheme for $L \in NP$ from the presented scheme.

2 Preliminaries

2.1 Notations and Assumptions.

Throughout the paper, λ is the security parameter. For any probabilistic polynomial time algorithm $A(\cdot)$, we use $y = A(x)$ or $y \leftarrow A(x)$ to denote the output of $A(x)$. For a set S, $y \leftarrow_R S$ denotes that y is uniformly chosen from S. For $L \in NP$, $\overline{L}$ is the complement of L. A function $negl(\cdot)$ is negligible if for any any polynomial $p(\cdot)$, $negl(\lambda) < 1/p(\lambda)$ when λ is large enough. We denote by $poly(\lambda)$ an arbitrary polynomial.

Average hard NP problem. Let L be an NP language with witness relation R_L. Assume there exists a sampling algorithm $\mathcal{S}$: $(x, w) \leftarrow \mathcal{S}(1^\lambda)$, where $|x| = \ell(\lambda), |w| = p(\lambda)$, such that $R_L(x, w) = 1$. Say that L is average hard if it is computationally impossible to distinguish between $x \in L$ and $x \in \overline{L}$. It is well known that the existence of average hard NP problem is equivalent to the existence of OWF. Here, for simplicity we assume that there exists a PPT sampling algorithm $\mathcal{S}'$ on $\overline{L}$, so L is average hard if for any PPT distinguisher $\mathcal{D}$, it holds that

$$\left| \Pr_{(x,w)\leftarrow\mathcal{S}(1^\lambda)} \left[\mathcal{D}(1^\lambda, x) = 1\right] - \Pr_{x\leftarrow\mathcal{S}'(1^\lambda)} \left[\mathcal{D}(1^\lambda, x) = 1\right] \right| = negl(\lambda)$$

Levin reduction. Levin reduction from NP language L with NP relation R_L to Γ with NP relation R_Γ consists of three computable functions π_0 and (π_1, π_1'), denoted as $\pi = (\pi_0, \pi_1, \pi_1')$, satisfying:

1) $y \in L \Leftrightarrow \pi_0(y) \in \Gamma$;
2) $R_L(y, w) = 1 \Rightarrow R_\Gamma(\pi_0(y), \pi_1(y, w)) = 1$;
3) $R_\Gamma(\pi_0(y), w') = 1 \Rightarrow R_L(y, \pi_1'(y, w')) = 1$.

where $|\pi_0(y)| = poly(|y|), |\pi_1(y, w)| = poly'(|y|)$ and $poly, poly'$ are any polynomials. It is known that most of the known NPC language have Levin reduction from any NP language.

2.2 Public Key Encryption

Definition 1. *An public key encryption scheme is a tuple of PPT algorithm $\Pi = (Gen, Enc, Dec)$,*

- *Key generation algorithm Gen: $(pk, sk) \leftarrow Gen(1^\lambda)$, Gen takes as input security parameter λ and outputs a pair of key (pk, sk);*

- *Probabilistic encryption algorithm* Enc*:* $c \leftarrow Enc(pk, m)$*,* Enc *takes as inputs public key* pk *and a message* $m \in M$ *and outputs ciphertext* c*;*
- *Decryption algorithm* Dec*:* $m' = Dec(sk, c)$*,* Dec *takes as input private key* sk *and a ciphertext* c *and outputs* m'*.*

And it satisfies the following properties:
Correctness*. For any message* m*, it is holds that*

$$\Pr[m = m' : (pk, sk) \leftarrow Gen(1^\lambda); c \leftarrow Enc(pk, m); m' = Dec(sk, c)] \geq 1 - negl(\lambda)$$

Chosen plaintext attack (CPA): Let $\Pi = (Gen, Enc, Dec)$ be PKE scheme. For any adversary $\mathcal{A}$, define CPA experiment $Expt^{\text{CPA}}_{\mathcal{A},\Pi}$ as follows:

$Expt^{\text{CPA}}_{\mathcal{A},\Pi}(1^\lambda, b)$:
- The challenger computes $(pk, sk) \leftarrow Gen(1^\lambda)$ and then sends pk to $\mathcal{A}$;
- $\mathcal{A}$ randomly selects two message m_0, m_1 from message space M and sends to challenger;
- The challenger computes $c \leftarrow Enc(pk, m_b)$ and sends c to $\mathcal{A}$;
- $\mathcal{A}$ returns b';
- $Expt^{\text{CPA}}_{\mathcal{A},\Pi}(1^\lambda) = b'$.

$\mathcal{A}$'s advantage is defined as follows:

$$Adv^{\text{CPA}}_{\mathcal{A},\Pi}(\lambda) = \left|\Pr\left[Expt^{\text{CPA}}_{\mathcal{A},\Pi}(1^\lambda, m_0) = 1\right] - \Pr\left[Expt^{\text{CPA}}_{\mathcal{A},\Pi}(1^\lambda, m_1) = 1\right]\right|$$

Specifically, if we focus on bit encryption, that is $M = \{0, 1\}$, the advantage of $\mathcal{A}$ is as follows:

$$Adv^{\text{CPA}}_{\mathcal{A},\Pi}(\lambda) = \left| \Pr_{\substack{pk, Enc \\ \mathcal{A}}}\left[\mathcal{A}(1^\lambda, pk, Enc(pk, 0)) = 1\right] - \Pr_{\substack{pk, Enc \\ \mathcal{A}}}\left[\mathcal{A}(1^\lambda, pk, Enc(pk, 1)) = 1\right] \right|$$

Definition 2. *(CPA secure) Assume* $\Pi = (Gen, Enc, Dec)$ *is a PKE scheme. Say* Π *is CPA secure, if for any PPT adversary* $\mathcal{A}$*,* $Adv^{CPA}_{\mathcal{A},\Pi}(\lambda)$ *is negligible.*

2.3 Witness Encryption

Definition 3. *([13]) An witness encryption scheme for an NP language* L *with NP relation* R *consists of the following two polynomial-time algotithms:*

- *Encryption algorithm* $Encrypt$*:* $c \leftarrow Encrypt(1^\lambda, x, m)$*,* $Encrypt$ *takes as input a security parameter* 1^λ*, an instance* x *and a message* $m \in M$*, and outputs ciphertext* c*;*
- *Decryption algorithm* $Decrypt$*:* $Decrypt(c, w) = m'/\perp$*,* Dec *takes as input a ciphertext* c *and an string* w*, and outputs* m' *or the symbol* $\perp$*.*

satisfying the following two conditions:

- ***Correctness****. For any message* m *and any* $x \in L$ *such that* $R(x, w) = 1$*, it is holds that*

$$\Pr[m = m' : c \leftarrow Encrypt(1^\lambda, x, m); m' = Decrypt(w, c)] \geq 1 - negl(\lambda)$$

- ***Soundness Security****. For any* $x \notin L$*, any PPT adversary* $\mathcal{A}$ *and two messages* m_0, m_1*, it is holds that*

$$\left|\Pr[\mathcal{A}(Encrypt(1^\lambda, x, m_0)) = 1] - \Pr[\mathcal{A}(Encrypt(1^\lambda, x, m_1)) = 1]\right| < negl(\lambda)$$

2.4 Garbled Circuits

In this section, we present a new garbled circuits scheme, known as garbled circuit scheme with constrained-input. Garbled circuit was first presented by Yao [23], and has been formalized by Bellare et al. [5]. Yao's garbled circuit is used for secure two-party computation in where both parties wish to carry out a joint computation of a shared function without disclosing their private input. The scenario and security requirements we consider in the paper are different from secure two-party computation.

Let C be a circuit that takes as input $(u, v) \in \{0,1\}^{\ell} \times \{0,1\}^{p}$ and outputs $z = C(u, v) \in \{0,1\}^{o}$. C is a directed acyclic graph that has $s = \ell + p$ input vertices (with no incoming edges) and o output vertices (with no outgoing edges and one incoming edge). All other vertices are logical operation gate, including OR gate, AND gate and NOT gate (corresponding to logical operator $\vee, \wedge, \neg$ respectively). We can assume, without loss of generality, that the outgoing edges of all NOT gates are only connected to AND or OR gates, and there is no output vertex whose incoming edge come from some input vertex.

The reason why we divide the input of C into two parts is that they are dealt with differently. In short, we will garble C for a given $\widetilde{u}$ such that one can obtain $C(\widetilde{u}, v)$ using the garbled circuit for any v. Specifically, for any given $\widetilde{u}$ we generate garbled circuit $\widehat{C}$ under constrained-input $u = \widetilde{u}$, and require that there exists an algorithm $Eval$ such that $Eval(\widehat{C}, v) = C(\widetilde{u}, v)$ for any v.

In this paper, the construction of circuit C is related to an average hard NP language. To be a little more specific, for an average hard NP language L with NP relation R_L and sampling algorithms $\mathcal{S}, \mathcal{S}'$, we construct a function $\Psi(u, v)$ by randomizing NP relation R_L, and then let C be a randomly selected circuit computing $\Psi(u, v)$, that is, $C(u, v) = \Psi(u, v)$. To garble C, we first determine an instance $\widetilde{x}$ and then obtain constrained-input $\widetilde{u}$. Garbled circuit $\widehat{C}$ under constrained-input $u = \widetilde{u}$ can be used to compute $\Psi(\widetilde{u}, v)$. To meet the need of constructing PKE scheme from average hard NP problem, we furthermore require that $\widehat{C}$ with constrained-input $\widetilde{u}$ can hide $\widetilde{u}$.

To achieve this aim, we need to formally eliminate all NOT gates in C by introducing composite gates. In detail, for any OR or AND gate g in C, assume that its two incoming edges are from vertices g' and g'' corresponding to the first and second inputs of g respectively, do as follows:

- If one of g' and g'' is NOT gate, such as g' (or g''), define composite logical g^*: $g^*(a, b) = g(\overline{a}, b)$ (or $g^*(a, b) = g(a, \overline{b})$), replace g with composite logical gate g^* and eliminate NOT gate g' (or g'') (see Fig. 1);
- If g' and g'' are all NOT gate, define composite logical g^*: $g^*(a, b) = g(\overline{a}, \overline{b})$ (in fact, if g is a OR or AND gate, then g^* is a NAND or NOR gate), replace g with composite logical gate g^* and eliminate NOT gates g' and g'' (see Fig. 2).

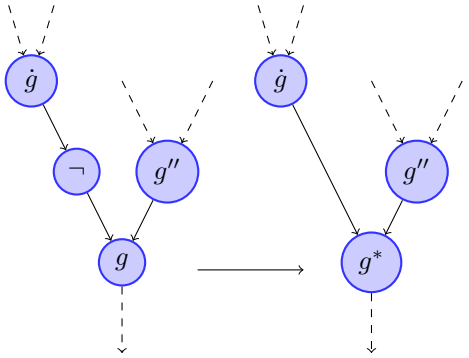

Fig. 1

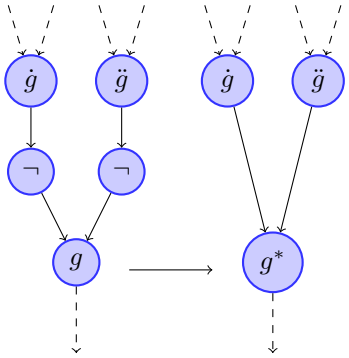

Fig. 2

Obviously, there are six different kinds of composite logic gates (including NAND, NOR), denoted as $\vee_1, \vee_2, \vee_3$ and $\wedge_1, \wedge_2, \wedge_3$, where

$$a \vee_1 b = \overline{a} \vee b \,, a \vee_2 b = a \vee \overline{b} \,, a \vee_3 b = \overline{a} \vee \overline{b}$$

$$a \wedge_1 b = \overline{a} \wedge b \,, a \wedge_2 b = a \wedge \overline{b} \,, a \wedge_3 b = \overline{a} \wedge \overline{b}$$

In the following, let's assume that the circuit C consists of 8 different logic gates ($\vee, \wedge$, and 6 composite gates). For simplicity, we use a symbol $\circledast$ to denote any composite logic gates.

2.4.1 Garbling Circuit Scheme

Let C be a circuit that takes as input $(u, v) \in \{0,1\}^{\ell} \times \{0,1\}^{p}$ and outputs $z = C(u, v) \in \{0,1\}^{o}$. C contains $s = \ell + p$ input vertices $V_1 = In_1, \cdots, V_s = In_s$, and t logical gate vertices $V_{s+1}, \cdots, V_{s+t}$ labeled logical operations $g_{s+1} \in \{\vee, \wedge, \circledast\}, \cdots, g_{s+t} \in \{\vee, \wedge, \circledast\}$ respectively, and o output vertices $V_{s+t+1} = Out_1, \cdots, V_{s+t+o} = Out_o$. On inputting $(u, v) \in \{0,1\}^{\ell} \times \{0,1\}^{p}$, C computes and outputs $C(u, v) \in \{0,1\}^{o}$. Following the ideal of Yao's garbled circuits, we define an algorithm $Garble$ to garble C with constrain-input $u = \widetilde{u}$, where $\widetilde{u} = \widetilde{u}_1 \cdots \widetilde{u}_{\ell} \in \{0,1\}^{\ell}$ is given.

Garbling algorithm: $Garble$

On inputting $(\widetilde{u}, C, \rho)$ and security parameter 1^{λ}, where $\rho \in \{0,1\}^{p}$, $Garble(\widetilde{u}, C, \rho; 1^{\lambda})$ proceeds as follows:

1. Randomly select $\sigma = \sigma_1 \cdots \sigma_p \in \{0,1\}^{p}$ and $s + t + o$ pairs of key $\{(K_i^0, K_i^1)\}_{i=1}^{s+t+o}$ (requiring $K_i^0 \neq K_i^1$), where $K_i^b \in \{0,1\}^{\lambda}$ is as the encoding of b ($b = 0, 1$). Assign $K_i^{\widetilde{u}_i}$ to input vertex In_i and its outgoing edge ($i = 1, \cdots, \ell$), and $(K_{\ell+j}^{\sigma_j}, K_{\ell+j}^{1-\sigma_j})$ to input vertex $In_{\ell+j}$ and its outgoing edge, $j = 1, \cdots, p$. $E = \left\{\left(K_{\ell+j}^0, K_{\ell+j}^1\right)\right\}_{j=1}^{p}$ is called the original encoding table and $D = \left\{\left(0, K_{s+t+i}^0\right), \left(1, K_{s+t+i}^1\right)\right\}_{i=1}^{o}$ is call the decoding table.
2. Generate all garbled logical gates, i.e. garbled computation tables for V_k ($k = s+1, \cdots, s+t$), gate by gate.

– For any $k \in \{s+1, \cdots, s+t\}$, assume V_k's two incoming edges are from V_i and V_j and V_i, V_j have been assigned. According to the assignment of V_i, V_j, it can be done as follows:
  - V_i is assigned K_i^a and V_j is assigned K_j^b. V_k's garbled computation table $\widehat{V_k}$ is as follows:

	V_i	V_j	V_k
1	K_i^a	K_j^b	$K_k^{g_k(a,b)}$

    and V_k and its outgoing edges are assigned $K_k^{g_k(a,b)}$;
  - V_i is assigned K_i^a and V_j is assigned (K_j^b, K_j^{1-b}). When $g_k(a,b) = g_k(a,1-b) = c$, V_k's garbled computation table $\widehat{V_k}$ is as follows:

	V_i	V_j	V_k
1	K_i^a	K_j^b	K_k^c
2	K_i^a	K_j^{1-b}	K_k^c

    and V_k with its outgoing edges are assigned K_k^c. When $c = g_k(a,b) \neq g_k(a,1-b)$, V_k's garbled computation table $\widehat{V_k}$ is as follows:

	V_i	V_j	V_k
1	K_i^a	K_j^b	K_k^c
2	K_i^a	K_j^{1-b}	K_k^{1-c}

    V_k and its outgoing edge are assigned (K_k^c, K_k^{1-c}), where $c = g_k(a,b)$;
  - V_i is assigned (K_i^a, K_i^{1-a}) and V_j is assigned K_j^b. When $g_k(a,b) = g_k(1-a,b) = c$, V_k's garbled computation table $\widehat{V_k}$ is as follows:

	V_i	V_j	V_k
1	K_i^a	K_j^b	K_k^c
2	K_i^{1-a}	K_j^b	K_k^c

    and V_k with its outgoing edges are assigned K_k^c. When $c = g_k(a,b) \neq g_k(1-a,b)$, V_k's garbled computation table $\widehat{V_k}$ is as follows:

	V_i	V_j	V_k
1	K_i^a	K_j^b	K_k^c
2	K_i^{1-a}	K_j^b	K_k^{1-c}

    V_k and its outgoing edge are assigned (K_k^c, K_k^{1-c}), where $c = g_k(a,b)$;
  - V_i is assigned (K_i^a, K_i^{1-a}) and V_j is assigned (K_j^b, K_j^{1-b}). V_k's garbled computation table $\widehat{V_k}$ is as follows:

	V_i	V_j	V_k
1	K_i^a	K_j^b	$K_k^{g_k(a,b)}$
2	K_i^a	K_j^{1-b}	$K_k^{g_k(a,1-b)}$
3	K_i^{1-a}	K_j^b	$K_k^{g_k(1-a,b)}$
4	K_i^{1-a}	K_j^{1-b}	$K_k^{g_k(1-a,1-b)}$

V_k and its outgoing edges are assigned (K_k^c, K_k^{1-c}), where $c = g_k(a, b)$.

– For $k \in \{s+t+1, \cdots, s+t+o\}$, V_k is output vertex. Assume V_k's incoming edges are from V_i. V_k is assigned K_k^a or (K_k^a, K_k^{1-a}) if V_i is assigned K_i^a or (K_i^a, K_i^{1-a}).

3. Use $K^{\widetilde{u}}$ to denote the encoding of $\widetilde{u}$, that is, $K^{\widetilde{u}} = \left(K_1^{\widetilde{u}_1}, \cdots, K_\ell^{\widetilde{u}_\ell}\right)$. Let

$$\widehat{E} = \left\{\left(\widehat{K}_{\ell+j}^0, \widehat{K}_{\ell+j}^1\right)\right\}_{j=1}^p = \left\{\left(K_{\ell+j}^{\rho_j}, K_{\ell+j}^{1-\rho_j}\right)\right\}_{j=1}^p$$

meaning $\widehat{K}_{\ell+j}^b$ $(j = 1, \cdots, p)$ will be treated as the encoding of b. $\widehat{E}$ is called the encoding table of $\widehat{C}$.

4. Output garbled circuit $\widehat{C} = \left\{K^{\widetilde{u}}, \widehat{E}, \{\widehat{V}_k\}_{k=s+1}^{s+t+o}, D\right\}$, that is

$$\widehat{C} = \left\{K^{\widetilde{u}}, \widehat{E}, \{\widehat{V}_k\}_{k=s+1}^{s+t}, D\right\} \leftarrow Garble(\widetilde{u}, C, \rho; 1^\lambda)$$

Note that our garbled circuits is slightly different in form from Yao's (see appendix A for details of Yao's garbled circuit). First of all, Yao's garbled circuits requires an encryption scheme to encrypt the garbled computation table, while our garbled circuits does not; Second, some garbled computation tables in garbled circuit $\widehat{C}$ are incomplete since the input vertex In_i is only assigned $K^{\widetilde{u}_i}$, $i = 1, \cdots, \ell$, that is, u is limited to be $\widetilde{u}$; Third, Yao's garbled circuit requires a private encoding table, and in our garbled scheme, although the encoding table $\widehat{E} = \left\{\left(\widehat{K}_{\ell+j}^0, \widehat{K}_{\ell+j}^1\right)\right\}_{j=1}^p$ is public, but the relationship between $\widehat{E}$ and the original encoding table E, i.e. the randomly selected $\rho = \rho_1 \cdots \rho_p$ implied by the relation $\left\{\left(\widehat{K}_{\ell+j}^0, \widehat{K}_{\ell+j}^1\right)\right\}_{j=1}^p = \left\{\left(K_{\ell+j}^{\rho_j}, K_{\ell+j}^{1-\rho_j}\right)\right\}_{j=1}^p$, is not public.

Let $\widehat{C} \leftarrow Garble(\widetilde{u}, C, \rho; 1^\lambda)$ (call it constrained-partial-input garbled circuit for C). $\widehat{C}$ is actually the encoded representation of $C(u, v)$ when fixing $u = \widetilde{u}$. So, $\widehat{C}$ can be used to compute $C(\widetilde{u}, v \oplus \rho)$ for any $v \in \{0,1\}^p$. To this end, we define the algorithm $Eval(\widehat{C}, w)$ which, on inputting $\widetilde{y}$ and any $v \in \{0,1\}^p$, compute $C(\widetilde{u}, w \oplus \rho)$. The algorithm $Eval(\widehat{C}, v)$ is follows:

$Eval(\widehat{C}, v)$

On inputting $w \in \{0,1\}^p$ and $\widehat{C}$, $Eval(\widehat{C}, v)$ completes:

1) Interpret $\widehat{C} = \left\{K^{\widetilde{u}}, \widehat{E}, \{\widehat{V}_k\}_{k=s+1}^{s+t+o}, D\right\}$, where $K^{\widetilde{u}} = \left(K_1^{\widetilde{u}_1}, \cdots, K_\ell^{\widetilde{u}_\ell}\right)$,

$$\widehat{E} = \left\{(\widehat{K}_{\ell+j}^0, \widehat{K}_{\ell+j}^1)\right\}_{j=1}^p, D = \left\{\left(0, K_{s+t+i}^0\right), \left(1, K_{s+t+i}^1\right)\right\}_{i=1}^o$$

2) Let $\widehat{K}^v = (\widehat{K}_{\ell+1}^{v_1}, \cdots, \widehat{K}_{\ell+p}^{v_p}) = (K_{\ell+1}^{v_1 \oplus \rho_1}, \cdots, K_{\ell+p}^{v_p \oplus \rho_p})$. ($\widehat{K}^v$ is the encoding of v corresponding to $\widehat{E}$)
3) Use $(K^{\widetilde{u}}, \widehat{K}^v)$ to compute all the garbled gates $V_{s+1}, \cdots, V_{s+t}$, gate by gate, by checking the corresponding garbled computation tables and then obtain the values assigned to the output vertices $V_{s+t+1}, \cdots, V_{s+t+o}$. Assume the value of the output vertex V_{s+t+i} is $K_{s+t+i}^{a_i}$, $i = 1, \cdots, o$.
4) Obtain $a_1, \cdots, a_o$ by checking D and let $Eval(w, \widehat{C}) = a_1 \cdots a_o$.

Obviously, since $\widehat{K}^v = (K_{\ell+1}^{v_1 \oplus \rho_1}, \cdots, K_{\ell+p}^{v_p \oplus \rho_p})$ is actually the encoding of $v \oplus \rho$, it holds that $Eval(\widehat{C}, v) = C(\widetilde{u}, v \oplus \rho)$. For convenience, we use $\widehat{C}(v)$ to denote $Eval(\widehat{C}, v)$, that is, $\widehat{C}(v) = Eval(\widehat{C}, v)$.

2.4.2 Hiding Property

Unlike Yao's garbled circuits used for secure two-party computation, our garbled circuits are used to build public-key encryption scheme, so the required security is also different from the security of Yao's garbled circuits, specifically, it requires that $\widehat{C} \leftarrow Garble(\widetilde{u}, C, \rho; 1^\lambda)$ can hide $\widetilde{u}$ and ρ. Obviously, if C is public, $\widehat{C} \leftarrow Garble(\widetilde{u}, C, \rho; 1^\lambda)$ is unlikely to hide $\widetilde{u}$ and ρ. However, when C is not public, that is, the truth tables of the logic gates in C are kept secret, $\widehat{C} \leftarrow Garble(\widetilde{u}, C, \rho; 1^\lambda)$ can hide $\widetilde{u}$ and ρ. In other words, it is impossible to obtain $\widetilde{u}$ and ρ from $\widehat{C} \leftarrow Garble(\widetilde{u}, C, \rho; 1^\lambda)$ without knowing C.

First, it is completely impossible to get $(\widetilde{u}, \rho)$ by computing $\widehat{C}(w) = Eval(\widehat{C}, w) = C(\widetilde{u}, w \oplus \rho)$, because $C(\widetilde{u}, v)$ doest not reveal any information about $(\widetilde{u}, v)$ when C is unknown.

Second, $\{\widehat{V}_k\}_{k=s+1}^{s+t+o}$ and D in $\widehat{C}$ (i.e. $\widehat{C} = \left\{K^{\widetilde{u}}, \widehat{E}, \{\widehat{V}_k\}_{k=s+1}^{s+t+o}, D\right\} \leftarrow Garble(\widetilde{u}, C, \rho; 1^\ell)$) do not reveal $\widetilde{u}$ and ρ. Clearly, $Garble$ "encrypts" the truth table of each logical gate g_k in C by encoding the inputs and output respectively as random keys, that is, the garbled logical gate for $\widehat{V}_k$ in $\widehat{C}$ is the encrypted truth table of g_k and is a garbled computation table. To indicate the relationship with g_k, we denote $\widehat{V}_k$ by $\widehat{g}_k$ below. There are six types of garbled computation tables, one is complete and five are incomplete. A complete garbled computation table $\widehat{g}_k$ is as follows:

Table 1 $\widehat{g}_k$

	V_i	V_j	V_k
1	K_i^a	K_j^b	$K_k^{c_1} = K_k^{g_k(a,b)}$
2	K_i^a	K_j^{1-b}	$K_k^{c_2} = K_k^{g_k(a,1-b)}$
3	K_i^{1-a}	K_j^b	$K_k^{c_3} = K_k^{g_k(1-a,b)}$
4	K_i^{1-a}	K_j^{1-b}	$K_k^{c_4} = K_k^{g_k(1-a,1-b)}$

An incomplete garbled computation table is a part of $\widehat{g}_k$ and has the following three forms:

	V_i	V_j	V_k
1	K_i^a	K_j^b	K_k^c

	V_i	V_j	V_k
1	K_i^a	K_j^b	K_k^c
2	K_i^{1-a}	K_j^b	K_k^c

	V_i	V_j	V_k
1	K_i^a	K_j^b	K_k^c
2	K_i^{1-a}	K_j^b	K_k^{1-c}

If each garbled computation table $\widehat{g}_k$ in $\widehat{C}$ hides the truth table of g_k, then $\widehat{C}$ does not leak $\widetilde{u}$ and ρ. Obviously, if any complete garbled computation table $\widehat{g}_k$ (as given in Table 1) does not leak the truth table of g_k (that is, a, b, c), then any incomplete garbled computation table will not. Therefore, to show $\widehat{C} \leftarrow Garble(\widetilde{u}, C, \rho; 1^\lambda)$ will not leak $\widetilde{u}$ and ρ, we only need to prove that any complete garbled computation table $\widehat{g}_k$ (as given in Table 1) in $\widehat{C}$ will not leak (a, b, c).

For a given $\widehat{g}_k$, there exist $c \in \{0,1\}$, such that only one of c_1, c_2, c_3 and c_4 equals c, the others are $1-c$. The fact that $\widehat{g}_k$ does not leak (a,b,c) means that a and b can be 1 or 0, respectively, regardless of which one of c_1, c_2, c_3, c_4 equals c and $c=0$ or $c=1$. Notice that $Garble$ picks randomly $\sigma = \sigma_1 \cdots \sigma_p$ and assigns $(K_{\ell+j}^{\sigma_j}, K_{\ell+j}^{1-\sigma_j})$ to $In_{\ell+j}$ $(j = 1, \cdots, p)$ and its outgoing edge. This means that the order of $\widehat{g}_k$ does not reveal a and b. So, the only thing that can be obtained from $\widehat{g}_k$ is which one of the c_1, c_2, c_3, c_4 is equal to c. This means that for a given $\widehat{g}_k$, g_k could be any one of the 8 possible logical gates ($\vee, \wedge$ and six composite logical gates), corresponding to 8 different sets of values for (a,b,c). In particular, if c is public, g_k can be any of the 4 possible logic gates (when $c=0$, the 4 possible logic gates are $\vee, \vee_1, \vee_2, \vee_3$; when $c=1$, the 4 possible logic gates are $\wedge, \wedge_1, \wedge_2, \wedge_3$), corresponding to 4 different sets of values for (a,b).

For example, when $\widehat{g}_k$ satisfies $c_2 = c$ and $c_1 = c_3 = c_4 = 1-c$, a and b can take any value, and different values correspond to different logic gates. Specifically, when $c=1$, the four different sets of values for (a,b) correspond four different (composite) logical gates, as shown in the following table (Table 2):

Table 2 Value of (a,b) and corresponding g_k

	a	b	g_k	$c_2 = g_k(a, 1-b)$	$c_1 = c_3 = c_4$
1	0	0	$g_k(a,b) = \overline{a} \wedge b$	1	0
2	0	1	$g_k(a,b) = \overline{a} \wedge \overline{b}$	1	0
3	1	0	$g_k(a,b) = a \wedge b$	1	0
4	1	1	$g_k(a,b) = a \wedge \overline{b}$	1	0

When $c=0$, the four different sets of values for (a,b) and the corresponding (composite) logical gate g_k are shown in the following table (Table 3):

Table 3 Value of (a,b) and corresponding g_k

	a	b	g_k	$c_2 = g_k(a, 1-b)$	$c_1 = c_3 = c_4$
1	0	0	$g_k(a,b) = a \vee \overline{b}$	0	1
2	0	1	$g_k(a,b) = a \vee b$	0	1
3	1	0	$g_k(a,b) = \overline{a} \vee \overline{b}$	0	1
4	1	1	$g_k(a,b) = \overline{a} \vee b$	0	1

$\widehat{C} \leftarrow Garble(\widetilde{u}, C, \rho; 1^\lambda)$ hides the constrained input $\widetilde{u}$ and the encoding of input v (i.e. ρ) when C is secret, but it reveals the topology (how gates are connected up) and size (number of gates) of C and have the functionality of C, that is, $\widehat{C}(w) = C(\widetilde{u}, w \oplus \rho)$ for any w. So, the following claims hold.

Claim 1 Let $C : \{0,1\}^{\ell+p} \to \{0,1\}^o$ and $u_0, u_1 \in \{0,1\}^\ell$. $\widehat{C}_\sigma \leftarrow Garble(\widetilde{u}_\sigma, C, \rho; 1^\lambda)$, $\sigma = 0, 1$. For any PPT algorithm $\mathcal{D}$, it holds that

$$\left| \Pr_{\mathcal{D}} \left[\mathcal{D}(u_0, u_1, \widehat{C}_\sigma) = \sigma \right] - \frac{1}{2} \right| = negl(\lambda) \tag{1}$$

That is, $(u_0, u_1, \widehat{C}_0)$ and $(u_0, u_1, \widehat{C}_1)$ are indistinguishable.

Proof. Let $s = \ell + p$ and assume that C has t logical gates. By the definition,

$$\widehat{C}_\sigma = \left\{K^{u_\sigma}, \widehat{E}, \{\widehat{V}_k\}_{k=s+1}^{s+t+o}, D\right\} \leftarrow Garble(u_\sigma, C, \rho; 1^\ell)$$

where

$$K^{u_\sigma} = \left(K_1^{u_{\sigma,1}}, \cdots, K_\ell^{u_{\sigma,\ell}}\right)$$
$$\widehat{E} = \left\{\left(\widehat{K}_{\ell+j}^0, \widehat{K}_{\ell+j}^1\right)\right\}_{j=1}^p = \left\{\left(K_{\ell+j}^{\rho_j}, K_{\ell+j}^{1-\rho_j}\right)\right\}_{j=1}^p$$

Let $\mu = \mu_1 \cdots \mu_\ell = u_0 \oplus u_1$. Define circuit C' by adding some "NOT" gates on C, specifically, adding a NOT gate ($\neg$) on the outgoing edge of the input vertex V_i when $\mu_i = 1$, $i = 1, \cdots, \ell$. Obviously, we have that

$$C'(u, v) = C(u \oplus \mu, v) \tag{2}$$

It is important to emphasize that C' and C are the same size and have the same topology (because of the use of composite logic gates). Furthermore, it follows from Equation (2) that if we reinterpret $K_i^{u_{\sigma,i}}$ (the encoding of $u_{\sigma,i}$) as $K_i^{u_{1-\sigma,i}}$ (the encoding of $u_{1-\sigma,i}$), $i = 1, \cdots, \ell$, (i.e. interpret K^{u_σ} as $K^{u_{1-\sigma}}$), then $\widehat{C}_\sigma$ is actually a garbled circuit of C' with the constrained-input $u = u_{1-\sigma}$.

In fact, interpreting K^{u_σ} as $K^{u_{1-\sigma}}$ is only related to the logic gates vertices where at least one input comes from the input nodes $V_1, \cdots, V_\ell$. Let V_k be the logical gate vertex, the corresponding logical gate is g_k, and the two inputs are from the nodes V_i and V_j. Here, we use $\widehat{g}_k$ to denote the garbled computation table $\widehat{V}_k$. When $i, j \le \ell$, V_i, V_j are assigned $K_i^{u_{\sigma,i}}, K_j^{u_{\sigma,j}}$, respectively, and the garbled computation table $\widehat{g}_k$ is shown in Table 4. When only one of i, j is greater than ℓ, for example $j > \ell$, V_i is assigned $K_i^{u_{\sigma,i}}$ and V_j is assigned $(K_j^{b_1}, K_j^{b_2})$ (where $b_1 \neq b_2$ or $b_1 = b_2$,), so $\widehat{g}_k$ is shown in Table 5 (when $b_1 = b_2$, ignore the second line):

Table 4 $\widehat{g}_k$ (when $i, j \le \ell$)

	V_i	V_j	V_k
1	$K_i^{u_{\sigma,i}}$	$K_j^{u_{\sigma,j}}$	$K_k^{c_1} = K_k^{g_k(u_{\sigma,i}, u_{\sigma,j})}$

Table 5 $\widehat{g}_k$ (when $i \le \ell, j > \ell$)

	V_i	V_j	V_k
1	$K_i^{u_{\sigma,i}}$	$K_j^{b_1}$	$K_k^{c_1} = K_k^{g_k(u_{\sigma,i}, b_1)}$
2	$K_i^{u_{\sigma,i}}$	$K_j^{b_2}$	$K_k^{c_2} = K_k^{g_k(u_{\sigma,i}, b_2)}$

(Here, $b_1 = b_2$ or $b_1 = 1 \oplus b_2$)

When $K^{u_{\sigma,i}}$ is reinterpreted as $K^{u_{1-\sigma,i}}$ and the output encoding $K_k^{c_1}$ remains unchanged (meaning that c_1, c_2 remain unchanged in Table 5), the garbled computation tables (Table 4 and Table 5) will be rewritten as follows, respectively:

Table 6 $\widehat{g}_k$ (when $i, j \le \ell$)
(replace $K_i^{u_{\sigma,i}}$ with $K_i^{u_{1-\sigma,i}}$)

	V_i	V_j	V_k
1	$K_i^{u_{1-\sigma,i}}$	$K_j^{u_{1-\sigma,j}}$	$K_k^{c_1} = K_k^{g_k(u_{1-\sigma,i} \oplus \mu_i, u_{1-\sigma,j} \oplus \mu_j)}$

Table 7 $\widehat{g}_k$ (when $i \le \ell, j > \ell$)
(replace $K_i^{u_{\sigma,i}}$ with $K_i^{u_{1-\sigma,i}}$)

	V_i	V_j	V_k
1	$K_i^{u_{1-\sigma,i}}$	$K_j^{b_1}$	$K_k^{c_1} = K_k^{g_k(u_{1-\sigma,i} \oplus \mu_i, b_1)}$
2	$K_i^{u_{1-\sigma,i}}$	$K_j^{b_2}$	$K_k^{c_2} = K_k^{g_k(u_{1-\sigma,i} \oplus \mu_i, b_2)}$

It is easy to see that after $K^{u_{\sigma,i}}$ is reinterpreted as $K^{u_{1-\sigma,i}}$ (while not changing the output of V_k), $\widehat{g}_k$ becomes the garbled computation table for g_k^* defined by $g_k^*(a,b) = g_k(a \oplus \mu_i, b)$. Therefore, when we reinterpret $K^{u_{\sigma,i}}$ as $K^{u_{1-\sigma,i}}$ for $i = 1, \cdots, \ell$, i.e. reinterpret K^{u_σ} as $K^{u_{1-\sigma}}$, $\widehat{C}_k$ is actually a garbled circuit of C' with the constrained-input $u = u_{1-\sigma}$. In other words, $\widehat{C}_\sigma$ can be reinterpreted as $\widehat{C}_\sigma \leftarrow Garble(\widetilde{u}_{1-\sigma}, C', \rho; 1^\lambda)$. It follows that getting σ from $(u_0, u_1, \widehat{C}_\sigma)$ is equivalent to guessing which of $C_\sigma \leftarrow Garble(u_\sigma, C, \rho; 1^\lambda)$ and $C_\sigma \leftarrow Garble(u_{1-\sigma}, C', \rho; 1^\lambda)$ holds, while when C is secret, the latter is obviously impossible (because C' and C are the same size and have the same topology). This argument implies that it must be impossible to get b only from $(u_0, u_1, \widehat{C}_b \leftarrow Garble(u_b, C, \rho; 1^\ell))$ when C is unknown. That is, Equation (1) holds. ∎

Claim 2 Let $C : \{0,1\}^{\ell+p} \to \{0,1\}^o$ and $\widetilde{u} \in \{0,1\}^\ell$. $\widehat{C}_\sigma \leftarrow Garble(\widetilde{u}, C, \rho^{(\sigma)}; 1^\lambda)$, where $\rho^{(\sigma)} = \rho_1^{(\sigma)} \cdots \rho_p^{(\sigma)} \in \{0,1\}^p$ is randomly selected, $\sigma = 0, 1$. For any PPT algorithm $\mathcal{D}$, it holds that

$$\left| \Pr_{\mathcal{D}} \left[\mathcal{D}(\rho^{(0)}, \rho^{(1)}, \widehat{C}_\sigma) = \sigma \right] - \frac{1}{2} \right| = negl(\lambda) \tag{3}$$

That is, $(\rho^{(0)}, \rho^{(1)}, \widehat{C}_0)$ and $(\rho^{(0)}, \rho^{(1)}, \widehat{C}_1)$ are indistinguishable.

Proof. Similar to the proof of Claim 1.

Let $s = \ell + p$ and assume that C has t logical gates. By the definition,

$$\widehat{C}_\sigma = \left\{ K^{u_\sigma}, \widehat{E}, \{\widehat{V}_k\}_{k=s+1}^{s+t+o}, D \right\} \leftarrow Garble(\widetilde{u}, C, \rho_\sigma; 1^\ell)$$

where

$$\widehat{E} = \left\{ \left(\widehat{K}_{\ell+j}^0, \widehat{K}_{\ell+j}^1 \right) \right\}_{j=1}^p = \left\{ \left(K_{\ell+j}^{\rho_j^{(\sigma)}}, K_{\ell+j}^{1-\rho_j^{(\sigma)}} \right) \right\}_{j=1}^p$$

Let $\delta = \delta_1 \cdots \delta_p = \rho^{(0)} \oplus \rho^{(1)}$. Define circuit C' as follows:

$$C'(u, v) = C(u, v \oplus \delta) \tag{4}$$

Obviously, C' can be obtained by adding some "NOT" gates on C, specifically, adding a NOT gate ($\neg$) on the outgoing edge of the input vertex $V_{\ell+j}$ when $\delta_j = 1$, $j = 1, \cdots, p$, so C' and C are the same size and have the same topology because composite logic gates are used. Furthermore, it follows that if we reinterpret $\left(\widehat{K}_{\ell+j}^0, \widehat{K}_{\ell+j}^1 \right) = \left(K_{\ell+j}^{\rho_j^{(\sigma)}}, K_{\ell+j}^{1-\rho_j^{(\sigma)}} \right)$ as $\left(\widehat{K}_{\ell+j}^0, \widehat{K}_{\ell+j}^1 \right) = \left(K_{\ell+j}^{\rho^{(1-\sigma)}}, K_{\ell+j}^{1-\rho_j^{(1-\sigma)}} \right)$ for $j = 1, \cdots, p$, then $\widehat{C}_\sigma$ is actually a garbled circuit of C' with the encoding table

$$\widehat{E} = \left\{ \left(\widehat{K}_{\ell+j}^0, \widehat{K}_{\ell+j}^1 \right) \right\}_{j=1}^p = \left\{ \left(K_{\ell+j}^{\rho_j^{(1-\sigma)}}, K_{\ell+j}^{1-\rho_j^{(1-\sigma)}} \right) \right\}_{j=1}^p$$

In fact, let V_k be the logical gate vertex, the corresponding logical gate is g_k, and the two inputs are from the nodes V_i and V_j. For convenience, we use $\widehat{g}_k$ to denote the garbled computation table $\widehat{V}_k$. Below, we only consider the case of $\ell < i, j \le \ell + p$, and it is completely similar when only one of the i, j is greater than ℓ but not more than $s = \ell + p$, so the details are omitted.

When both V_i and V_j are input vertices and $i, j > \ell$, the garbled computation table $\widehat{g}_k$ in $\widehat{C}_\sigma$ for g_k is shown in Table 8:

Table 8 $\widehat{g}_k$

	V_i	V_j	V_k
1	$K_i^{\rho_i^{(\sigma)}}$	$K_j^{\rho_j^{(\sigma)}}$	$K_k^{c_1} = K_k^{g_k\left(\rho_i^{(\sigma)}, \rho_j^{(\sigma)}\right)}$
2	$K_i^{\rho_i^{(\sigma)}}$	$K_j^{1-\rho_j^{(\sigma)}}$	$K_k^{c_2} = K_k^{g_k\left(\rho_i^{(\sigma)}, 1-\rho_j^{(\sigma)}\right)}$
3	$K_i^{1-\rho_i^{(\sigma)}}$	$K_j^{\rho_j^{(\sigma)}}$	$K_k^{c_3} = K_k^{g_k\left(1-\rho_i^{(\sigma)}, \rho_j^{(\sigma)}\right)}$
4	$K_i^{1-\rho_i^{(\sigma)}}$	$K_j^{1-\rho_j^{(\sigma)}}$	$K_k^{c_4} = K_k^{g_k\left(1-\rho_i^{(\sigma)}, 1-\rho_j^{(\sigma)}\right)}$

When we replace $\left(K_{\ell+i}^{\rho_i^{(\sigma)}}, K_{\ell+i}^{1-\rho_i^{(\sigma)}}\right)$ with $\left(K_{\ell+i}^{\rho_i^{(1-\sigma)}}, K_{\ell+j}^{1-\rho_i^{(1-\sigma)}}\right)$ and do not change $K_k^{c_1}, K_k^{c_2}, K_k^{c_3}, K_k^{c_4}$, $\widehat{g}_k$ will be rewritten as follows and denoted as $\widehat{g}_k^*$:

Table 9 $\widehat{g}_k^*$

	V_i	V_j	V_k
1	$K_i^{\rho_i^{(1-\sigma)}}$	$K_j^{\rho_j^{(\sigma)}}$	$K_k^{c_1} = K_k^{g_k\left(\rho_i^{(1-\sigma)} \oplus \delta_i, \rho_j^{(\sigma)}\right)}$
2	$K_i^{\rho_i^{(1-\sigma)}}$	$K_j^{1-\rho_j^{(\sigma)}}$	$K_k^{c_2} = K_k^{g_k\left(\rho_i^{(1-\sigma)} \oplus \delta_i, 1-\rho_j^{(\sigma)}\right)}$
3	$K_i^{1-\rho_i^{(1-\sigma)}}$	$K_j^{\rho_j^{(\sigma)}}$	$K_k^{c_3} = K_k^{g_k\left(1-\rho_i^{(1-\sigma)} \oplus \delta_i, \rho_j^{(\sigma)}\right)}$
4	$K_i^{1-\rho_i^{(1-\sigma)}}$	$K_j^{1-\rho_j^{(\sigma)}}$	$K_k^{c_4} = K_k^{g_k\left(1-\rho_i^{(1-\sigma)} \oplus \delta_i, 1-\rho_j^{(\sigma)}\right)}$

It is easy to see that $\widehat{g}_k^*$ (rewritten $\widehat{g}_k$) is the the garbled computation table for g_k^* defined by $g_k^*(a, b) = g_k(a \oplus \delta_i, b)$. In other words, if we reinterpret $\left(\widehat{K}_{\ell+i}^0, \widehat{K}_{\ell+i}^1\right) = \left(K_{\ell+i}^{\rho_i^{(\sigma)}}, K_{\ell+i}^{1-\rho_i^{(\sigma)}}\right)$ as $\left(\widehat{K}_{\ell+i}^0, \widehat{K}_{\ell+i}^1\right) = \left(K_{\ell+i}^{\rho_i^{(1-\sigma)}}, K_{\ell+i}^{1-\rho_i^{(1-\sigma)}}\right)$, the garbled computation table $\widehat{g}_k$ becomes the garbled computation table for g_k^*.

Similarly, if we reinterpret $\left(K_{\ell+i}^{\rho_i^{(\sigma)}}, K_{\ell+i}^{1-\rho_i^{(\sigma)}}\right)$ as $\left(K_{\ell+i}^{\rho_i^{(1-\sigma)}}, K_{\ell+i}^{1-\rho_i^{(1-\sigma)}}\right)$ and $\left(K_{\ell+j}^{\rho_j^{(\sigma)}}, K_{\ell+j}^{1-\rho_j^{(\sigma)}}\right)$ as $\left(K_{\ell+j}^{\rho_j^{(1-\sigma)}}, K_{\ell+j}^{1-\rho_j^{(1-\sigma)}}\right)$, the garbled computation table $\widehat{g}_k$ becomes the garbled computation table for g_k^* defined by $g_k^*(a, b) = g_k(a \oplus \delta_i, b \oplus \delta_j)$.

Therefore, if we reinterpret $\left(\widehat{K}_{\ell+j}^0, \widehat{K}_{\ell+j}^1\right) = \left(K_{\ell+j}^{\rho_j^{(\sigma)}}, K_{\ell+j}^{1-\rho_j^{(\sigma)}}\right)$ as $\left(\widehat{K}_{\ell+j}^0, \widehat{K}_{\ell+j}^1\right) = \left(K_{\ell+j}^{\rho^{(1-\sigma)}}, K_{\ell+j}^{1-\rho_j^{(1-\sigma)}}\right)$ for $j = 1, \cdots, p$, then $\widehat{C}_\sigma$ is actually a garbled circuit of C' with the encoding table

$$\widehat{E} = \left\{\left(\widehat{K}_{\ell+j}^0, \widehat{K}_{\ell+j}^1\right)\right\}_{j=1}^p = \left\{\left(K_{\ell+j}^{\rho_j^{(1-\sigma)}}, K_{\ell+j}^{1-\rho_j^{(1-\sigma)}}\right)\right\}_{j=1}^p$$

In other words, $\widehat{C}_\sigma$ can be interpreted as $C_\sigma \leftarrow Garble(u, C', \rho^{(1-\sigma)}; 1^\lambda)$. Therefore, extracting σ from $(\rho^{(0)}, \rho^{(1)}, \widehat{C}_\sigma)$ is equivalent to guessing which of $C_\sigma \leftarrow Garble(u, C, \rho^{(\sigma)}; 1^\lambda)$ and $C_\sigma \leftarrow Garble(u, C', \rho^{(1-\sigma)}; 1^\lambda)$ holds, while the latter is impossible when C is secret, so extracting σ from $(\rho^{(0)}, \rho^{(1)}, \widehat{C}_\sigma)$ is also impossible. That is, Equation (3) holds. ∎

Claim 3 Let $C : \{0,1\}^{\ell+p} \to \{0,1\}^o$ and $u_0, u_1 \in \{0,1\}^\ell$. $\widehat{C}_\sigma \leftarrow Garble(u_\sigma, C, \rho^{(\sigma)}; 1^\lambda)$, where $\rho^{(\sigma)} = \rho_1^{(\sigma)} \cdots \rho_p^{(\sigma)}$ is randomly selected, $\sigma = 0, 1$. For any PPT algorithm $\mathcal{D}$, it holds that

$$\left| \Pr_{\mathcal{D}} \left[\mathcal{D}((u_0, \rho^{(0)}), (u_1, \rho^{(1)}), \widehat{C}_\sigma) = \sigma \right] - \frac{1}{2} \right| = negl(\lambda) \tag{5}$$

That is, $((u_0, \rho^{(0)}), (u_1, \rho^{(1)}), \widehat{C}_0)$ and $((u_0, \rho^{(0)}), (u_1, \rho^{(1)}), \widehat{C}_1)$ are indistinguishable.

Proof. Similar to the proofs of Claim 1 and Claim 2.

Let $s = \ell + p$ and assume that C has t logical gates. By the definition,

$$\widehat{C}_\sigma = \left\{ K^{u_\sigma}, \widehat{E}, \{\widehat{V}_k\}_{k=s+1}^{s+t+o}, D \right\} \leftarrow Garble(\widetilde{u}, C, \rho_\sigma; 1^\ell)$$

where

$$K^{u_\sigma} = \left(K_1^{u_{\sigma,1}}, \cdots, K_\ell^{u_{\sigma,\ell}} \right)$$

$$\widehat{E} = \left\{ \left(\widehat{K}_{\ell+j}^0, \widehat{K}_{\ell+j}^1 \right) \right\}_{j=1}^p = \left\{ \left(K_{\ell+j}^{\rho_j^{(\sigma)}}, K_{\ell+j}^{1-\rho_j^{(\sigma)}} \right) \right\}_{j=1}^p$$

Let $\mu = \mu_1 \cdots \mu_\ell = u_0 \oplus u_1, \delta = \delta_1 \cdots \delta_p = \rho^{(0)} \oplus \rho^{(1)}$. Define circuit C' as follows:

$$C'(u, v) = C(u \oplus \mu, v \oplus \delta) \tag{6}$$

Obviously, C' can be obtained by adding some "NOT" gates on the outgoing edge of the input vertex of C, so C' and C are the same size and have the same topology.

From the proofs of Claim 1 and Claim 2, it follows that if we reinterpret $K^{u_{\sigma,i}}$ as $K^{u_{1-\sigma,i}}$ for $i = 1, \cdots, \ell$, and reinterpret $\left(\widehat{K}_{\ell+j}^0, \widehat{K}_{\ell+j}^1 \right) = \left(K_{\ell+j}^{\rho_j^{(\sigma)}}, K_{\ell+j}^{1-\rho_j^{(\sigma)}} \right)$ as $\left(\widehat{K}_{\ell+j}^0, \widehat{K}_{\ell+j}^1 \right) = \left(K_{\ell+j}^{\rho_j^{(1-\sigma)}}, K_{\ell+j}^{1-\rho_j^{(1-\sigma)}} \right)$ for $j = 1, \cdots, p$, then $\widehat{C}_\sigma$ is actually a garbled circuit of C' with the constrained-input $u_{1-\sigma}$ and the encoding table

$$\widehat{E} = \left\{ \left(\widehat{K}_{\ell+j}^0, \widehat{K}_{\ell+j}^1 \right) \right\}_{j=1}^p = \left\{ \left(K_{\ell+j}^{\rho_j^{(1-\sigma)}}, K_{\ell+j}^{1-\rho_j^{(1-\sigma)}} \right) \right\}_{j=1}^p$$

In other words, $\widehat{C}_\sigma$ can be interpreted as $C_\sigma \leftarrow Garble(u_{1-\sigma}, C', \rho^{(1-\sigma)}; 1^\lambda)$. Therefore, getting σ from $((u_0, \rho^{(0)}), (u_1, \rho^{(1)}), \widehat{C}_\sigma)$ is equivalent to guessing which of $C_\sigma \leftarrow Garble(u_\sigma, C, \rho^{(\sigma)}; 1^\lambda)$ and $C_\sigma \leftarrow Garble(u_{1-\sigma}, C', \rho^{(1-\sigma)}; 1^\lambda)$ holds, while the latter is impossible when C is secret. Therefore, getting σ from $((u_0, \rho^{(0)}), (u_1, \rho^{(1)}), \widehat{C}_\sigma)$ is also impossible, that is, Equation (5) holds. The claim thus follows. ∎

3 PKE from Average Hard Language

We focus on bit encryption scheme, since the existence of one bit encryption is sufficient to construct PKE schemes for messages of arbitrary length.

3.1 Basic Idea

Let L be an average hard NP language with NP relation R_L and sampling algorithms $\mathcal{S}$ on L. PKE schemes based on L use an instance $x \in L$ and its corresponding witness w as public key and private key, respectively. Before giving the presented schemes, we first sketch the basic idea of constructing PKE scheme.

Let $(x, w) \leftarrow \mathcal{S}(1^\lambda)$. Clearly, L is average hard means that to search a witness w for $x \in L$ (satisfying $R_L(x, w) = 1$) must be hard. This further implies that, if only black-box access is allowed, no PPT algorithm (without holding x's witness) can distinguish $f_1(u) = R(x, u)$ and $f_0(u) \equiv 0$. Therefore, if we can construct circuits, denoted as $\widehat{C}_b$ ($b = 0, 1$), from x, such that $\widehat{C}_b(u) = f_b(u)$ and obtaining $\widehat{C}_b$ is equivalent to black-box access to $\widehat{C}_b$, then $\widehat{C}_b$ can be used as the ciphertext of b. With this motivation in the mind, we can naturally recall VBB obfuscator defined by Barak in [1]. Specifically, if there exists an VBB obfuscator, denoted by $\mathcal{O}$, PKE scheme based on L is very simple: for $(x, w) \leftarrow \mathcal{S}(1^\lambda)$, set $(pk, sk) = (x, w)$, compute $c_b = \mathcal{O}(f_b)$ when encrypting b and compute $b' = c_b(sk)$ when decrypting c_b. Clearly, the correctness and security of the scheme are respectively derived from the functionality and VBB property of $\mathcal{O}$.

Unfortunately, VBB obfuscator is too strong and does not exist even if OWF exists. But based on this observation, we can get a new strategy of constructing PKE from average hard NP language L, described as follows:

> *For $(x, w) \leftarrow \mathcal{S}(1^\lambda)$, construct random (garbled) circuits $\widehat{C}_b$ ($b = 0, 1$) just based on the average hardness of L, satisfying the following conditions:*
> 1) *When $R_L(x, w) = 1$, $\widehat{C}_b(w) = b$ (correctness);*
> 2) *The circuit $\widehat{C}_b$ itself will not reveal b, that is, $\widehat{C}_0$ and $\widehat{C}_1$ are computationally indistinguishable (security).*

Next, we will show how to generate such (garbled) circuits $\widehat{C}_b$ for $(x, w) \leftarrow \mathcal{S}(1^\lambda)$. Briefly, it can be divided into two steps:

First, we construct a random function Ψ by randomizing NP relation R_L, and then select a circuit C computing Ψ. Specifically, for $(x, w) \leftarrow \mathcal{S}(1^\lambda)$, where $|x| = \ell, |w| = p$, we randomly select $\alpha \in \{0,1\}^\ell, \beta \in \{0,1\}^p$ and then define a random algorithm $Randomize$ to generate circuit C. The details of $Randomize$ is as follows.

$$Randomize(R_L, \alpha, \beta; 1^\lambda)$$

- Randomly select one-to-one function F, where $F : \{0,1\}^\ell \to \{0,1\}^{\ell'}$, $\ell' \geq \ell$.
- Define $L' \in NP$ as follows:

$$L' = \{y : \exists u, v, \text{ such that } y = F(u), R_L(u \oplus \alpha, v \oplus \beta) = 1\}$$

  Let $R_{L'}$ be the corresponding NP relation.
- Randomly select NPC language Γ with NP relation R_Γ, and let $\pi = (\pi_0, \pi_1, \pi_1')$ be Levin reduction from L' to Γ. Consider the following function:

$$\Psi(u, v) = R_\Gamma(\pi_0(F(u)), \pi_1(F(u), (u, v)))$$

- Select circuit C computing $\Psi(u, v)$, that is, $C(u, v) = \Psi(u, v)$.
- Output C.

For a given function Ψ, there are many circuits computing Ψ. If C to compute Ψ, it is not difficult to obtain a new circuit, which is functionally equivalent to C, from C. For Example, let C computes Ψ, we can first construct a circuit C' equivalent to C by adding a pair of NOT gates on the one incoming edge of any logic gate in C (since $\neg\neg a = a$), and then apply De Morgan's law to convert C' into its equivalent circuit C'', requiring C'' satisfies the following requirement: the outgoing edges of all NOT gates are only connected to AND or OR gates, and there is no output vertex whose incoming edge come from some input vertex. Obviously, C'' is functionally equivalent to C.

It is easy to see that $\Psi(u, v)$ is closely related to R_L. Specifically, since $\pi = (\pi_0, \pi_1, \pi_1')$ is Levin reduction from L' to Γ, $\Psi(u, v)$ satisfies the following three properties:

1) When $R_L(x, w) = 1$, there must be $\Psi(u, v) = 1$ for $u = x \oplus \alpha, v = w \oplus \beta$.
   (In fact, $R_L(x, w) = 1$ means that $(x \oplus \alpha, w \oplus \beta)$ is a witness of $y = F(x \oplus \alpha) \in L'$. So, by the Levin reduction π, $\Psi(x \oplus \alpha, w \oplus \beta) = 1$.)
2) When $\Psi(u, v) = 0$, $R_L(x, w) = 0$ holds for $x = u \oplus \alpha, w = v \oplus \beta$.
   (In fact, although $\Psi(u, v) = 0$ does not implies $\pi_0(F(u)) \notin \Gamma$, it means that $\pi_1(F(u), (u, v))$ is not a witness for $\pi_0(F(u)) \in \Gamma$. This further means that (u, v) must not be a witness for $F(u) \in L'$, that is, $R_L(x, w) = 0$ for $x = u \oplus \alpha$ and $w = v \oplus \beta$.)
3) When $\Psi(u, v) = 1$, there must be $x = u \oplus \alpha \in L$.
   (In fact, $\Psi(u, v) = 1$ means that $\pi_0(F(u)) \in \Gamma$, so $F(u) \in L'$. That is, there exist v such that $R_L(u \oplus \alpha, v \oplus \beta) = 1$. This means $x = u \oplus \alpha \in L$.)

Second, we construct $\widehat{C}_b$ by a garbling algorithm. We briefly show how to generate $\widehat{C}_b$ for the given public key $pk = (x, \lambda)$: $(x, w) \leftarrow \mathcal{S}(1^\lambda)$, $|x| = \ell, |w| = p$. Assume that $C \leftarrow Randomize(R_L, \alpha, \beta; 1^\lambda)$, where α, β is randomly selected. C contains $s = \ell' + p$ input vertices $V_1, \cdots, V_{\ell'+p}$, and t composite logical gate vertices $V_{s+1}, \cdots, V_{s+t}$ labeled logical operations $g_{s+1}, \cdots, g_{s+t}$, and one output vertices V_{s+t+1}. We can obtain $\widehat{C}_b$ by garbling $C(u, v)$ with constrain-input $u = \widetilde{u}$.

Let $GenGC$ be the algorithm generating the garbled circuits $\widehat{C}_b$, the details of are as follows:

$$GenGC(x; 1^\lambda)$$

- Randomly select $x_0 \in \overline{L}$: $x_0 \leftarrow \mathcal{S}'(1^\lambda)$. Set $x_1 = x$.
- Randomly select $\alpha \in \{0,1\}^\ell, \beta \in \{0,1\}^p$.
- Let $C \leftarrow Randomize(R_L, \alpha, \beta; 1^\lambda)$.
- Compute $\widetilde{u} = x_b \oplus \alpha$, and then garble $C(u, v)$ with constraining $u = \widetilde{u}$. That is, let

$$\widehat{C}_b = \left\{ K^{\widetilde{u}}, \widehat{E}, \left\{ \widehat{V}_k \right\}_{k=s+1}^{s+t+1}, D \right\} \leftarrow Garble\left(\widetilde{u}, C, \beta; 1^\lambda\right)$$

  where $D = \left\{ \left(0, K^0_{s+t+1}\right), \left(1, K^1_{s+t+1}\right) \right\}$ and

$$K^{\widetilde{u}} = \left\{ K_i^{\widetilde{u}_i} \right\}_{i=1}^{\ell},\ \widehat{E} = \left\{ \left(\widehat{K}^0_{\ell+j}, \widehat{K}^1_{\ell+j} \right) \right\}_{j=1}^{p} = \left\{ \left(K^{\beta_j}_{\ell+j}, K^{1-\beta_j}_{\ell+j} \right) \right\}_{j=1}^{p}$$

- Output $\widehat{C}_b$.

Here, for simplicity we assume that $\mathcal{S}'(1^\lambda)$ is distributed on $\overline{L}$, that is, $\Pr[\mathcal{S}'(1^\lambda) \in \overline{L}] = 1$. We show that $\widehat{C}_b$ generated by $GenGC$ meets the two requirements mentioned above.

1) $\widehat{C}_b(w)$ reveals b when $R_L(x, w) = 1$. In fact, from the construction of Ψ and the correctness of $Garble$, we have that

$$\begin{aligned}\widehat{C}_b(w) &= Eval(\widehat{C}_b, w)\\ &= C(\widetilde{u}, w \oplus \beta)\\ &= C(F(x_b \oplus \alpha), w \oplus \beta)\\ &= R_\Gamma(\pi_0(F(x_b \oplus \alpha)), \pi_1(F(x_b \oplus \alpha), x_b \oplus \alpha, w \oplus \beta))\end{aligned}$$

So, when $R_L(x_1, w) = 1$, $\widehat{C}_1(w) = 1$ since $\widetilde{y} = Fun(x_1 \oplus \alpha) \in L'$ and $R_{L'}(\widetilde{y}, (x_b \oplus \alpha, w \oplus \beta)) = 1$. In addition, $x_0 \in \overline{L}$ means $F(x_0 \oplus \alpha) \notin L'$ because F is one-to-one, thus also means $\pi_0(F(x_b \oplus \alpha) \notin \Gamma$, so $\widehat{C}_0(v) = 0$ for any v.

2) $\widehat{C}_b$ itself does not reveal b. In fact, $\widehat{C}_b$ and $\widehat{C}_{1-b}$ are computationally indistinguishable, and the proofs for this are given in Subsection 3.2 (i.e. the proof of theorem 1). Here we only show that it is impossible to get b by computing C_b. When $R_L(x, w) = 0$, $\widehat{C}_0(w) = \widehat{C}_1(w) \equiv 0$, it means that it is impossible to get b by computing $\widehat{C}_b$ unless there is w satisfying $R_L(x, w) = 1$. However, the assumption that L is average hard ensures that no adversary can find w from $\widehat{C}_1$ such that $\widehat{C}_1(w) = 1$.

Assume, on the contrary, that there exists a PPT algorithm $\mathcal{A}$ which can find w, satisfying $\widehat{C}_1(w) = 1$, with a non-negligible probability ε, that is,

$$\Pr[\widehat{C}_1(w) = 1 : x \leftarrow \mathcal{S}(1^\lambda), \widehat{C}_1 \leftarrow GenGC(x; 1^\lambda), w \leftarrow \mathcal{A}(x, \widehat{C}_1; 1^\lambda)] = \varepsilon$$

we can define PPT algorithm $\mathcal{B}$ which can find witness for $x \in L$ with the same probability ε. The details of $\mathcal{B}$ are as follows:

$\mathcal{B}(x; 1^\lambda)$

On inputting x and security parameter λ, $\mathcal{B}(x; 1^\lambda)$ do:

- Randomly select $\alpha \in \{0,1\}^\ell, \beta \in \{0,1\}^p$.
- Run $Randomize(R_L, \alpha, \beta; 1^\lambda)$, record F (one-to-one function) and π (Levin reduction from L' to Γ) and finally obtain $C \leftarrow Randomize(R_L, \alpha, \beta; 1^\lambda)$;
- Compute $\widetilde{u} = x \oplus \alpha$, and then garble $C(u, v)$ with constrain-input $u = \widetilde{u}$, i.e.

$$\widehat{C}_1 \leftarrow Garble\left(\widetilde{u}, C, \beta; 1^\lambda\right)$$

- Run $\mathcal{A}(x, \widehat{C}_1; 1^\lambda)$ and obtain its return w: $w \leftarrow \mathcal{A}(x, \widehat{C}_1; 1^\lambda)$;
- If $w = \perp$ or $\widehat{C}_1(w) = 0$, fail and stop;
- If $\widehat{C}_1(w) = 1$, do as follows:
  - Compute $\phi = \pi_1(F(\widetilde{u}), \widetilde{u}, w \oplus \beta)$;
  - Set $(u', v') = \pi_1'(\widetilde{u}, \phi)$, where $|u'| = \ell, |v'| = p$.
- Output $v = v' \oplus \beta$.

Let $\widetilde{u} = x \oplus \alpha$. Obviously, when w (the return of $\mathcal{A}(\widehat{C}_1)$) satisfies $\widehat{C}_1(w) = 1$, that is,

$$R_\Gamma(\pi_0(F(\widetilde{u})), \pi_1(F(\widetilde{u}), \widetilde{u}, w \oplus \beta)) = 1$$

This means that $\phi = \pi_1(F(\widetilde{u}), \widetilde{u}, w \oplus \beta)$ is the witness for $z = \pi_0(F(\widetilde{u})) \in \Gamma$. From the definition of Levin reduction π, it follows that $(u', v') = \pi'_1(\widetilde{y}, \phi)$ is a witness for $\widetilde{y} = F(\widetilde{u}) \in L'$, that is, (u', v') satisfies $\widetilde{y} = F(u')$ and $R_L(u' \oplus \alpha, v' \oplus \beta) = 1$. Notice that $u' = \widetilde{u} = x \oplus \alpha$ because F is one-to-one. So, $R_L(x, v' \oplus \beta) = R_L(u' \oplus \alpha, v' \oplus \beta) = 1$. In other words, $v' \oplus \beta$ is the witness for $x \in L$. Therefore, we obtain

$$\Pr[R_L(x, v) = 1 : x \leftarrow \mathcal{S}(1^\lambda), v \leftarrow \mathcal{B}(x; 1^\lambda)] = \varepsilon$$

This contradicts the assumption that L is average hard.

3.2 PKE Scheme I

3.2.1 Construction of PKE Scheme

Assume that L is an average hard NP language with NP relation R_L and sampling algorithms $\mathcal{S}$ on L and $\mathcal{S}'$ on $\overline{L}$. In fact, it is sufficient for our PKE scheme to require $\Pr[\mathcal{S}'(1^\lambda) \notin \overline{L}] \leq negl(\lambda)$.

Our tool for constructing PKE schemes is garbled circuits with constrained-input, and the security of PKE schemes depends on the hiding property of garbled circuits. Our first PKE scheme $\Pi_1 = (Gen, Enc, Dec)$ as follows:

Construction 1: PKE scheme $\Pi_1 = (Gen, Enc, Dec)$

- $Gen(1^\lambda)$: $(x, w) \leftarrow \mathcal{S}(1^\lambda)$, set $pk = (x, \lambda)$ and $sk = w$. $|x| = \ell, |w| = p$.
- $Enc(pk, b)$, $b \in \{0, 1\}$:
  - Randomly select $\alpha \in \{0,1\}^\ell$ by selecting $\overline{x} \in \overline{L}$ (i.e. $\overline{x} \leftarrow \mathcal{S}'(1^\lambda)$) and setting $\alpha = x \oplus \overline{x}$. Randomly select $\beta = \beta_1 \cdots \beta_p \in \{0,1\}^p$.
  - Run $Randomize$ to get a circuit C: $C \leftarrow Randomize(R_L, \alpha, \beta; 1^\lambda)$, where $C : \{0,1\}^\ell \times \{0,1\}^p \to \{0,1\}$. Assume C contains t logical gate vertices and set $s = \ell + p$.
  - If $b = 1$, set $\widetilde{u} = x \oplus \alpha$; otherwise set $\widetilde{u} = x' \oplus \alpha$, where $x' \leftarrow \mathcal{S}'(1^\lambda)$.
  - Garble C with constraining $u = \widetilde{u}$, i.e.

$$\widehat{C} = \left\{ K^{\widetilde{y}}, \widehat{E}, \left\{ \widehat{V}_k \right\}_{k=s+1}^{s+t+1}, D \right\} \leftarrow Garble\left(\widetilde{u}, C, \beta; 1^\lambda\right)$$

    where

$$K^{\widetilde{u}} = \{K_i^{\widetilde{u}_i}\}_{i=1}^{\ell},\ D = \left\{\left(0, K_{s+t+1}^0\right), \left(1, K_{s+t+1}^1\right)\right\}$$

$$\widehat{E} = \left\{ \left(\widehat{K}_{\ell+j}^0, \widehat{K}_{\ell+j}^1 \right) \right\}_{j=1}^{p} = \left\{ \left(K_{\ell+j}^{\beta_j}, K_{\ell+j}^{1-\beta_j} \right) \right\}_{j=1}^{p}$$

  - Output ciphertext $c_b = \widehat{C}$.
- $Dec(sk, c_b)$: Interpret $c_b = \widehat{C} = \left\{ K^{\widetilde{u}}, \widehat{E}, \left\{ \widehat{V}_k \right\}_{k=s+1}^{s+t+1}, D \right\}$ as a garbled circuit. Compute $a = Dec(sk, c_b) = \widehat{C}(sk) = Eval(sk, \widehat{C})$ and output a.

Correctness. Obviously, if $\Pr[\mathcal{S}'(1^\lambda) \in \overline{L}] = 1$, it holds that

$$\Pr[Dec(sk, Enc(pk, b)) = b : (pk, sk) \leftarrow Gen(1^\lambda)] = 1$$

In fact, by the definition,

$$a = Dec(sk, Enc(pk, b)) = R_\Gamma(\pi_0(F(x_b \oplus \alpha)), \pi_1(F(x_b \oplus \alpha), x_b \oplus \alpha, sk \oplus \beta))$$

where $x_1 = x \in L, x_0 = x' \in \overline{L}$. When $b = 1$, $(x_1 \oplus \alpha, sk \oplus \beta)$ is a witness for $F(x_1 \oplus \alpha) \in L'$, where L' is defined as follows:

$$L' = \{y : \exists u, v, \text{ such that } y = F(u), R_L(u \oplus \alpha, v \oplus \beta) = 1\}$$

So, $\pi_0(F(x_1 \oplus \alpha)) \in \Gamma$ and the corresponding witness is $\pi_1(F(x_1 \oplus \alpha), x_1 \oplus \alpha, sk \oplus \beta)$ since π is Levin reduction from L' to Γ, that is, $a = Dec(sk, Enc(pk, 1)) = 1$. When $b = 0$, by the definition, we have that $F(x_0 \oplus \alpha) \notin L'$. It follows that $\pi_0(F(x_0 \oplus \alpha)) \notin \Gamma$. Therefore, we further get that

$$a = Dec(sk, Enc(pk, 0)) = R_\Gamma(\pi_0(F(x_0 \oplus \alpha)), \pi_1(F(x_0 \oplus \alpha), x_b \oplus \alpha, sk \oplus \beta)) = 0$$

Obviously, if $\mathcal{S}'$ satisfies $\Pr[\mathcal{S}'(1^\lambda) \notin \overline{L}] \leq negl(\lambda)$, an error may occur only when decrypting the ciphertext of 0, and the error probability must be negligible.

3.2.2 Proof of Security

In the following, we assume that $\Pr[\mathcal{S}'(1^\lambda) \in \overline{L}] = 1$ for simplicity. We start from proving the following lemmas.

Lemma 1. *For any $x, x' \leftarrow \mathcal{S}'(1^\lambda)$, let $\widehat{C}' \leftarrow Enc(pk', 1)$, where $pk' = (x', \lambda)$, then $(x, \widehat{C}')$ and $(x', \widehat{C}')$ are computationally indistinguishable. That is, for any PPT algorithm $\mathcal{A}$, it holds that*

$$\left| \Pr_{\substack{x' \leftarrow \mathcal{S}'(1^\lambda) \\ x \leftarrow \mathcal{S}'(1^\lambda) \\ Enc, \mathcal{A}}} [\mathcal{A}(x, \widehat{C}') = 1] - \Pr_{\substack{x' \leftarrow \mathcal{S}'(1^\lambda) \\ Enc, \mathcal{A}}} [\mathcal{A}(x', \widehat{C}') = 1] \right| = negl(\lambda) \tag{7}$$

Proof. Let $\widehat{C} \leftarrow Enc(pk, 1)$, where $pk = (x, \lambda)$. It is easy to see that $(x, \widehat{C})$ and $(x', \widehat{C}')$ have the same distribution. So, we only need to show that $(x, \widehat{C})$ and $(x, \widehat{C}')$ are computationally indistinguishable.

By the definition, we have that

$$\widehat{C} \leftarrow Garble\left(\widetilde{u}, C, \beta; 1^\lambda\right), \widetilde{u} = x \oplus \alpha$$

$$\widehat{C}' \leftarrow Garble\left(\widetilde{u}', C, \beta; 1^\lambda\right), \widetilde{u}' = x' \oplus \alpha$$

where α, β are randomly selected and $C \leftarrow Randomize(R_L, \alpha, \beta; 1^\lambda)$. Note that C computes $\Psi(u, v) = R_\Gamma(\pi_0(F(u)), \pi_1(F(u), (u, v)))$, where $\pi = (\pi_0, \pi_1, \pi_1')$ is Levin reduction from L':

$$L' = \{y : \exists u, v, \text{ such that } y = F(u), R_L(u \oplus \alpha, v \oplus \beta) = 1\}$$

to Γ with NP relation R_Γ.

To prove $(x, \widehat{C})$ and $(x, \widehat{C}')$ are computationally indistinguishable, we let $u_0 = x$ and

$$\widehat{C}_0 \leftarrow Garble\left(u_0, C, \beta; 1^\lambda\right)$$

and then prove that both $(x, \widehat{C})$ and $(x, \widehat{C}')$ are computationally indistinguishable from $(x, \widehat{C}_0)$.

1) $(x, \widehat{C}_0)$ and $(x, \widehat{C}')$ are computationally indistinguishable. Since $x, \overline{x} \notin L$, it is easy to see that $C(u_0, v) = C(\widetilde{u} \cdot v) \equiv 0$ for any $v \in \{0,1\}^p$. Therefore, by Claim 1 (given in Subsection 2.4), we have that $(u_0, \widetilde{u}', \widehat{C}_0)$ and $(u_0, \widetilde{u}', \widehat{C}')$ are computationally indistinguishable. It follows that $(x, \widehat{C}_0) = (u_0, \widehat{C}_0)$ and $(x, \widehat{C}') = (u_0, \widehat{C}')$ are computationally indistinguishable.
2) $(x, \widehat{C}_0)$ and $(x, \widehat{C})$ are computationally indistinguishable. Similar to 1), by Claim 1 (given in Subsection 2.4), we have that $(u_0, \widetilde{u}, \widehat{C}_0)$ and $(u_0, \widetilde{u}, \widehat{C})$ are computationally indistinguishable. So, $(u_0, \widehat{C}_0)$ and $(u_0, \widehat{C})$ are computationally indistinguishable, i.e. $(x, \widehat{C}_0)$ and $(x, \widehat{C})$ are computationally indistinguishable.

Combining 1) and 2), we have that $(x, \widehat{C})$ and $(x, \widehat{C}')$ are computationally indistinguishable. It follows that $(x, \widehat{C}')$ and $(x', \widehat{C}')$ are computationally indistinguishable. This means that Equation (7) holds. ∎

Lemma 2. *For any $x \leftarrow \mathcal{S}(1^\lambda), x' \leftarrow \mathcal{S}'(1^\lambda)$, let $\widehat{C}' = Enc(pk', 1)$, where $pk' = (x', \lambda)$, then $(x, \widehat{C}')$ and $(x', \widehat{C}')$ are computationally indistinguishable. That is, for any PPT algorithm $\mathcal{A}$, we have*

$$\left| \Pr_{\substack{x' \leftarrow \mathcal{S}'(1^\lambda) \\ x \leftarrow \mathcal{S}(1^\lambda) \\ Enc, \mathcal{A}}} [\mathcal{A}(x, \widehat{C}') = 1] - \Pr_{\substack{x' \leftarrow \mathcal{S}'(1^\lambda) \\ Enc, \mathcal{A}}} [\mathcal{A}(x', \widehat{C}') = 1] \right| = negl(\lambda) \tag{8}$$

Proof. Assume towards contradiction that there exists a PPT algorithm $\mathcal{A}$ distinguishing $(x, \widehat{C}')$ and $(x', \widehat{C}')$ with advantage $\varepsilon(\lambda) \geq \frac{1}{poly(\lambda)}$ for positive polynomial $poly$. That is,

$$\left| \Pr_{\substack{x' \leftarrow \mathcal{S}'(1^\lambda) \\ x \leftarrow \mathcal{S}(1^\lambda) \\ Enc, \mathcal{A}}} [\mathcal{A}(x, \widehat{C}') = 1] - \Pr_{\substack{x' \leftarrow \mathcal{S}'(1^\lambda) \\ Enc, \mathcal{A}}} [\mathcal{A}(x', \widehat{C}') = 1] \right| = \varepsilon(\lambda) \geq \frac{1}{poly(\lambda)}$$

holds for infinitely λ's.

To obtain a contradiction, we construct an algorithm $\mathcal{D}$ to distinguish $x \in L$ and $x \in \overline{L}$ as follows:

$\mathcal{D}(x; 1^\lambda)$

On inputting x and security parameter λ, $\mathcal{D}(x; 1^\lambda)$ do:

- Let $x' \leftarrow \mathcal{S}'(1^\lambda)$, set $pk' = (x', \lambda)$.
- Run $Enc(pk', 1)$ and obtain $\widehat{C}' \leftarrow Enc(pk', 1)$.
- Randomly select $b \in \{0, 1\}$.
- If $b = 0$, invoke $\mathcal{A}$ with $\Delta \stackrel{def}{=} (x', \widehat{C}')$, otherwise invoke $\mathcal{A}$ with $\Delta \stackrel{def}{=} (x, \widehat{C}')$. Finally, obtain$\mathcal{A}$'s return b': $b' \leftarrow \mathcal{A}(\Delta)$.
- Output 1 if $b' = b$, otherwise, output 0.

Then we have

$$\begin{aligned}\Pr_{\substack{x\leftarrow\mathcal{S}(1^\lambda)\\ \mathcal{D}}}[\mathcal{D}(x;1^\lambda)=1] =& \frac{1}{2}\Pr_{\substack{x'\leftarrow\mathcal{S}'(1^\lambda)\\ x\leftarrow\mathcal{S}(1^\lambda)\\ Enc,\mathcal{A}}}[\mathcal{A}(x,\widehat{C}')=1]+\frac{1}{2}\Pr_{\substack{x'\leftarrow\mathcal{S}'(1^\lambda)\\ Enc,\mathcal{A}}}[\mathcal{A}(x',\widehat{C}')=0]\\ =&\frac{1}{2}+\frac{1}{2}\Pr_{\substack{x'\leftarrow\mathcal{S}'(1^\lambda)\\ x\leftarrow\mathcal{S}(1^\lambda)\\ Enc,\mathcal{A}}}[\mathcal{A}(x,\widehat{C}')=1]-\frac{1}{2}\Pr_{\substack{x'\leftarrow\mathcal{S}'(1^\lambda)\\ Enc,\mathcal{A}}}[\mathcal{A}(x',\widehat{C}')=1] \quad (9)\end{aligned}$$

and

$$\begin{aligned}\Pr_{\substack{x\leftarrow\mathcal{S}'(1^\lambda)\\ \mathcal{D}}}[\mathcal{D}(x;1^\lambda)=1] =& \frac{1}{2}\Pr_{\substack{x'\leftarrow\mathcal{S}'(1^\lambda)\\ x\leftarrow\mathcal{S}'(1^\lambda)\\ Enc,\mathcal{A}}}[\mathcal{A}(x,\widehat{C}')=1]+\frac{1}{2}\Pr_{\substack{x'\leftarrow\mathcal{S}'(1^\lambda)\\ Enc,\mathcal{A}}}[\mathcal{A}(x',\widehat{C}')=0]\\ =&\frac{1}{2}+\frac{1}{2}\Pr_{\substack{x'\leftarrow\mathcal{S}'(1^\lambda)\\ x\leftarrow\mathcal{S}'(1^\lambda)\\ Enc,\mathcal{A}}}[\mathcal{A}(x,\widehat{C}')=1]-\frac{1}{2}\Pr_{\substack{x'\leftarrow\mathcal{S}'(1^\lambda)\\ Enc,\mathcal{A}}}[\mathcal{A}(x',\widehat{C}')=1]\end{aligned} \quad (10)$$

Combining Equation (9) with Equation (10), it follows that

$$\begin{aligned}&\left|\Pr_{\substack{x\leftarrow\mathcal{S}(1^\lambda)\\ \mathcal{D}}}[\mathcal{D}(x;1^\lambda)=1]-\Pr_{\substack{x\leftarrow\mathcal{S}'(1^\lambda)\\ \mathcal{D}}}[\mathcal{D}(x;1^\lambda)=1]\right|\\ =&\frac{1}{2}\left|\Pr_{\substack{x'\leftarrow\mathcal{S}'(1^\lambda)\\ x\leftarrow\mathcal{S}(1^\lambda)\\ Enc,\mathcal{A}}}[\mathcal{A}(x,\widehat{C}')=1]-\Pr_{\substack{x'\leftarrow\mathcal{S}'(1^\lambda)\\ Enc,\mathcal{A}}}[\mathcal{A}(x',\widehat{C}')=1]\right.\\ &\left.-\Pr_{\substack{x'\leftarrow\mathcal{S}'(1^\lambda)\\ x\leftarrow\mathcal{S}'(1^\lambda)\\ Enc,\mathcal{A}}}[\mathcal{A}(x,\widehat{C}')=1]+\Pr_{\substack{x'\leftarrow\mathcal{S}'(1^\lambda)\\ Enc,\mathcal{A}}}[\mathcal{A}(x',\widehat{C}')=1]\right|\\ \geq&\frac{1}{2}\varepsilon-\frac{1}{2}\left|\Pr_{\substack{x'\leftarrow\mathcal{S}'(1^\lambda)\\ x\leftarrow\mathcal{S}'(1^\lambda)\\ Enc,\mathcal{A}}}[\mathcal{A}(x,\widehat{C}')=1]-\Pr_{\substack{x'\leftarrow\mathcal{S}'(1^\lambda)\\ Enc,\mathcal{A}}}[\mathcal{A}(x',\widehat{C}')=1]\right|\end{aligned}$$

Furthermore, by Lemma 1, we have that

$$\left|\Pr_{\substack{x\leftarrow\mathcal{S}(1^\lambda)\\ \mathcal{D}}}[\mathcal{D}(x;1^\lambda)=1]-\Pr_{\substack{x\leftarrow\mathcal{S}'(1^\lambda)\\ \mathcal{D}}}[\mathcal{D}(x;1^\lambda)=1]\right|\geq\frac{1}{2}\left(\varepsilon(\lambda)-negl(\lambda)\right) \quad (11)$$

for infinitely λ's. Equation (11) contradicts the assumption that L is an average hard NP language. So Equation (8) holds. ∎

Lemma 3. *For any $x\leftarrow\mathcal{S}(1^\lambda), x'\leftarrow\mathcal{S}(1^\lambda)$, let $\widehat{C}'=Enc(pk',1)$, where $pk'=(x',\lambda)$, then $((x,\lambda),\widehat{C}')$ and $((x',\lambda),\widehat{C}')$ are computationally indistinguishable. That is, for any* PPT

algorithm $\mathcal{A}$, it holds that

$$\left| \Pr_{\substack{x' \leftarrow \mathcal{S}(1^\lambda) \\ Enc, \mathcal{A}}} [\mathcal{A}((x', \lambda), \widehat{C}') = 1] - \Pr_{\substack{x' \leftarrow \mathcal{S}(1^\lambda) \\ x \leftarrow \mathcal{S}(1^\lambda) \\ Enc, \mathcal{A}}} [\mathcal{A}((x, \lambda), \widehat{C}') = 1] \right| = negl(\lambda) \tag{12}$$

Proof. Obviously, we only need to prove $(x, \widehat{C}')$ and $(x', \widehat{C}')$ are computationally indistinguishable. Assume, on the contrary, that there exists a PPT algorithm $\mathcal{A}$ which can distinguish $(x, \widehat{C}')$ and $(x', \widehat{C}')$ with advantage $\varepsilon \geq \frac{1}{poly(\lambda)}$ for positive polynomial $poly$, that is,

$$\left| \Pr_{\substack{x' \leftarrow \mathcal{S}(1^\lambda) \\ Enc, \mathcal{A}}} [\mathcal{A}(x', \widehat{C}') = 1] - \Pr_{\substack{x' \leftarrow \mathcal{S}(1^\lambda) \\ x \leftarrow \mathcal{S}(1^\lambda) \\ Enc, \mathcal{A}}} [\mathcal{A}(x, \widehat{C}') = 1] \right| = \varepsilon(\lambda) \geq \frac{1}{poly(\lambda)}$$

holds for infinitely λ's.

To obtain a contradiction, we construct an algorithm $\mathcal{D}$ to distinguish $x' \in L$ from $x' \in \overline{L}$. The details of $\mathcal{D}$ are as follows:

$\mathcal{D}(x'; 1^\lambda)$

On inputting x' and security parameter λ, $\mathcal{D}(x'; 1^\lambda)$ do:

- Set $pk' = (x', \lambda)$.
- Run $Enc(pk', 1)$ and obtain $\widehat{C}' \leftarrow Enc(pk', 1)$.
- Randomly select $b \in \{0, 1\}$, $x \leftarrow \mathcal{S}(1^\lambda)$.
- If $b = 0$, invoke $\mathcal{A}$ with $\Delta \stackrel{def}{=} (x, \widehat{C}')$, otherwise invoke $\mathcal{A}$ with $\Delta \stackrel{def}{=} (x', \widehat{C}')$. Finally, obtain$\mathcal{A}$'s return b': $b' \leftarrow \mathcal{A}(\Delta)$.
- Output 1 if $b' = b$, otherwise, output 0.

From the definition of $\mathcal{D}$, we have

$$\begin{aligned} \Pr_{\substack{x' \leftarrow \mathcal{S}(1^\lambda) \\ \mathcal{D}}} [\mathcal{D}(x'; 1^\lambda) = 1] &= \frac{1}{2} \Pr_{\substack{x' \leftarrow \mathcal{S}(1^\lambda) \\ Enc, \mathcal{A}}} [\mathcal{A}(x', \widehat{C}') = 1] + \frac{1}{2} \Pr_{\substack{x' \leftarrow \mathcal{S}(1^\lambda) \\ x \leftarrow \mathcal{S}(1^\lambda) \\ Enc, \mathcal{A}}} [\mathcal{A}(x, \widehat{C}') = 0] \\ &= \frac{1}{2} + \frac{1}{2} \Pr_{\substack{x' \leftarrow \mathcal{S}(1^\lambda) \\ Enc, \mathcal{A}}} [\mathcal{A}(x', \widehat{C}') = 1] - \frac{1}{2} \Pr_{\substack{x' \leftarrow \mathcal{S}(1^\lambda) \\ x \leftarrow \mathcal{S}(1^\lambda) \\ Enc, \mathcal{A}}} [\mathcal{A}(x, \widehat{C}') = 1] \end{aligned} \tag{13}$$

and

$$\begin{aligned} \Pr_{\substack{x' \leftarrow \mathcal{S}'(1^\lambda) \\ \mathcal{D}}} [\mathcal{D}(x'; 1^\lambda) = 1] &= \frac{1}{2} \Pr_{\substack{x' \leftarrow \mathcal{S}'(1^\lambda) \\ Enc, \mathcal{A}}} [\mathcal{A}(x', \widehat{C}') = 1] + \frac{1}{2} \Pr_{\substack{x' \leftarrow \mathcal{S}'(1^\lambda) \\ x \leftarrow \mathcal{S}(1^\lambda) \\ Enc, \mathcal{A}}} [\mathcal{A}(x, \widehat{C}') = 0] \\ &= \frac{1}{2} + \frac{1}{2} \Pr_{\substack{x' \leftarrow \mathcal{S}'(1^\lambda) \\ Enc, \mathcal{A}}} [\mathcal{A}(x', \widehat{C}') = 1] - \frac{1}{2} \Pr_{\substack{x' \leftarrow \mathcal{S}'(1^\lambda) \\ x \leftarrow \mathcal{S}(1^\lambda) \\ Enc, \mathcal{A}}} [\mathcal{A}(x, \widehat{C}') = 1] \end{aligned} \tag{14}$$

Combining Equation (13) with Equation (14), we obtain that

$$\begin{aligned}
&\left|\Pr_{\substack{x'\leftarrow\mathcal{S}(1^\lambda)\\ \mathcal{D}}}[\mathcal{D}(x',1^\lambda)=1]-\Pr_{\substack{x'\leftarrow\mathcal{S}'(1^\lambda)\\ \mathcal{D}}}[\mathcal{D}(x',1^\lambda)=1]\right| \\
=&\frac{1}{2}\left|\Pr_{\substack{x'\leftarrow\mathcal{S}(1^\lambda)\\ Enc,\mathcal{A}}}[\mathcal{A}(x',\widehat{C}')=1]-\frac{1}{2}\Pr_{\substack{x'\leftarrow\mathcal{S}(1^\lambda)\\ x\leftarrow\mathcal{S}(1^\lambda)\\ Enc,\mathcal{A}}}[\mathcal{A}(x,\widehat{C}')=1]\right. \\
&\left.+\frac{1}{2}\Pr_{\substack{x'\leftarrow\mathcal{S}'(1^\lambda)\\ x\leftarrow\mathcal{S}(1^\lambda)\\ Enc,\mathcal{A}}}[\mathcal{A}(x,\widehat{C}')=1]-\frac{1}{2}\Pr_{\substack{x'\leftarrow\mathcal{S}'(1^\lambda)\\ Enc,\mathcal{A}}}[\mathcal{A}(x',\widehat{C}')=1]\right| \\
\geq&\frac{1}{2}\varepsilon(\lambda)-\frac{1}{2}\left|\Pr_{\substack{x'\leftarrow\mathcal{S}'(1^\lambda)\\ x\leftarrow\mathcal{S}(1^\lambda)\\ Enc,\mathcal{A}}}[\mathcal{A}(x,\widehat{C}')=1]-\Pr_{\substack{x'\leftarrow\mathcal{S}'(1^\lambda)\\ Enc,\mathcal{A}}}[\mathcal{A}(x',\widehat{C}')=1]\right|
\end{aligned}$$

Moreover, by Lemma 2, it holds that

$$\left|\Pr_{\substack{x'\leftarrow\mathcal{S}'(1^\lambda)\\ x\leftarrow\mathcal{S}(1^\lambda)\\ Enc,\mathcal{A}}}[\mathcal{A}(x,\widehat{C}')=1]-\Pr_{\substack{x'\leftarrow\mathcal{S}'(1^\lambda)\\ Enc,\mathcal{A}}}[\mathcal{A}(x',\widehat{C}')=1]\right|=negl(\lambda)$$

Therefore, we have that

$$\left|\Pr_{\substack{x'\leftarrow\mathcal{S}(1^\lambda)\\ \mathcal{D}}}[\mathcal{D}(x',1^\lambda)=1]-\Pr_{\substack{x'\leftarrow\mathcal{S}'(1^\lambda)\\ \mathcal{D}}}[\mathcal{D}(1^\lambda,x')=1]\right|\geq\frac{1}{2}\left(\varepsilon(\lambda)-negl(\lambda)\right) \tag{15}$$

holds for infinitely λ's. This contradicts the assumption that L is an average hard NP language. The lemma thus follows. ∎

From the above lemmas, we can prove the following theorem:

Theorem 1. *The PKE scheme Π_1 is CPA secure when L is an average hard NP language.*

Proof. Assume $(x,w)\leftarrow\mathcal{S}(1^\lambda)$, where $|x|=\ell,|w|=p$, $pk=(x,\lambda)$ and $sk=w$. Let $c_b\leftarrow Enc(pk,b)$.

Note that c_0 and c_1 are constructed from $x'\notin L$ and $x\in L$ respectively, so they are indistinguishable because L is average hard. Briefly, according to the definition of Enc, c_b is a garbled circuit, $c_b=\widehat{C}$, and so, to distinguish (pk,c_0) and (pk,c_1) means to determine whether there exists w such that $\widehat{C}(w)=1$. This must be difficult without knowing w which satisfies $R_L(x,w)=1$.

Suppose, to the contrary, Π is not CPA secure, that is, there is a CPA adversary $\mathcal{A}$ and a polynomial $poly$, such that

$$Adv^{\text{CPA}}_{\mathcal{A},\Pi_1}(\lambda)=\left|\Pr_{\substack{pk,Enc\\ \mathcal{A}}}[\mathcal{A}(pk,Enc(pk,0))=1]-\Pr_{\substack{pk,Enc\\ \mathcal{A}}}[\mathcal{A}(pk,Enc(pk,1))=1]\right|>\frac{1}{poly(\lambda)}$$

holds for infinitely λ's. Then, we can construct an algorithm $\mathcal{D}$ to distinguish $x' \in L$ from $x' \in \overline{L}$ with a non-negligible probability.

Define the algorithm $\mathcal{D}$ as follows:

$\mathcal{D}(x'; 1^\lambda)$

On inputting x' and security parameter λ, $\mathcal{D}(x'; 1^\lambda)$ do:

- Let $(x, w) \leftarrow \mathcal{S}(1^\lambda)$, set $pk = (x, \lambda)$. $|x| = \ell, |w| = p$.
- Run Enc without specifying plaintext to get a ciphertext $\widehat{C}$, denoted by $\widehat{C} \leftarrow Enc(pk, ?)$. The details are as follows:
  - Randomly select $\alpha \in \{0,1\}^\ell, \beta \in \{0,1\}^p$.
  - $C \leftarrow Randomize(R_L, \alpha, \beta; 1^\lambda)$. Assume C has t logical gates.
  - Set $\widetilde{u} = x' \oplus \alpha$.
  - Garble C with constraining $u = \widetilde{u}$, i.e.

$$\widehat{C} = \left\{ K^{\widetilde{u}}, \widehat{E}, \left\{ \widehat{V}_k \right\}_{k=s+1}^{s+t+1}, D \right\} \leftarrow Garble\left(\widetilde{u}, C, \beta; 1^\lambda\right)$$

  where $s = \ell + p$.
- Invoke $\mathcal{A}$ with $(pk, \widehat{C})$ and obtain $\mathcal{A}$'s return b': $b' \leftarrow \mathcal{A}(pk, \widehat{C})$.
- Output b'.

Obviously, when $x' \in \overline{L}$, $\widehat{C} \leftarrow Enc(pk, ?)$ is the same as $\widehat{C} \leftarrow Enc(pk, 0)$, that is, $\widehat{C}$ is a normal ciphertext of 0 (with $pk = (x, \lambda)$ as the public key). So it holds that

$$\Pr_{\substack{x \leftarrow \mathcal{S}'(1^\lambda) \\ \mathcal{D}}} [\mathcal{D}(x'; 1^\lambda) = 1] = \Pr_{\substack{pk, Enc \\ \mathcal{A}}} [\mathcal{A}(pk, Enc(pk, 0)) = 1] \tag{16}$$

In contrast, when $x' \in L$, $\widehat{C}$, generated by $\widehat{C} \leftarrow Enc(pk, ?)$, is not a normal ciphertext obtained by encrypting 1 with $pk = (x, \lambda)$ because $\widetilde{u} = x' \oplus \alpha$ is inconsistent with $pk = (x, \lambda)$. However, by Lemma 3, we have that $(x, \widehat{C})$ and $(x', \widehat{C})$ are computationally indistinguishable. That is, for any PPT algorithm $\mathcal{B}$, it is holds that

$$\left| \Pr_{\substack{\widehat{C}, \mathcal{B} \\ x \leftarrow \mathcal{S}(1^\lambda)}} \left[\mathcal{B}((x, \lambda), \widehat{C}) = 1 \right] - \Pr_{\substack{\widehat{C}, \mathcal{B} \\ x' \leftarrow \mathcal{S}'(1^\lambda)}} \left[\mathcal{B}((x', \lambda), \widehat{C}) = 1 \right] \right| = negl(\lambda) \tag{17}$$

Furthermore, note that, when $x' \in L$, $\widehat{C} \leftarrow Enc(pk, ?)$ is the same as $\widehat{C} \leftarrow Enc(pk', 1)$, i.e. if $pk' = (x', \lambda)$ is used as public key, then $\widehat{C}$ is a normal ciphertext of 1. Therefore, we have that

$$\begin{aligned} \Pr_{\substack{\mathcal{B}, \widehat{C} \\ x' \leftarrow \mathcal{S}(1^\lambda)}} \left[\mathcal{B}((x', \lambda), \widehat{C}) = 1 \right] &= \Pr_{\substack{\mathcal{B}, Enc \\ (pk', sk') \leftarrow Gen(1^\lambda)}} [\mathcal{B}(pk', Enc(pk', 1)) = 1] \\ &= \Pr_{\substack{\mathcal{B}, Enc \\ (pk, sk) \leftarrow Gen(1^\lambda)}} [\mathcal{B}(pk, Enc(pk, 1)) = 1] \end{aligned} \tag{18}$$

Since Equations (17) holds for $\mathcal{A}$, so, from Equation (16),(17), we have that, for infinitely λ's,

$$\begin{aligned}
&\left|\Pr_{(x',w')\leftarrow\mathcal{S}(1^\lambda)}\left[\mathcal{D}(x';1^\lambda)=1\right]-\Pr_{x'\leftarrow\mathcal{S}'(1^\lambda)}\left[\mathcal{D}(x';1^\lambda)=1\right]\right| \\
=&\left|\Pr_{\substack{(x',w')\leftarrow\mathcal{S}(1^\lambda)\\(x,w)\leftarrow\mathcal{S}(1^\lambda)\\\mathcal{A},\widehat{C}}}\left[\mathcal{A}((x,\lambda),\widehat{C})=1\right]-\Pr_{\substack{x'\leftarrow\mathcal{S}'(1^\lambda)\\(x,w)\leftarrow\mathcal{S}(1^\lambda)\\\mathcal{A},\widehat{C}}}\left[\mathcal{A}((x,\lambda),\widehat{C})=1\right]\right| \\
=&\left|\Pr_{\substack{(x',w')\leftarrow\mathcal{S}(1^\lambda)\\\mathcal{A},\widehat{C}}}\left[\mathcal{A}((x',\lambda),\widehat{C})=1\right]\pm negl(\lambda)-\Pr_{\substack{x'\leftarrow\mathcal{S}'(1^\lambda)\\(x,w)\leftarrow\mathcal{S}(1^\lambda)\\\mathcal{A},\widehat{C}}}\left[\mathcal{A}((x,\lambda),\widehat{C})=1\right]\right|
\end{aligned}$$

Notice that (18) holds for $\mathcal{A}$, that is,

$$\Pr_{\substack{\mathcal{A},\widehat{C}\\x'\leftarrow\mathcal{S}(1^\lambda)}}\left[\mathcal{A}((x',\lambda),\widehat{C})=1\right]=\Pr_{\substack{\mathcal{A},Enc\\(pk',sk')\leftarrow Gen(1^\lambda)}}\left[\mathcal{A}(pk',Enc(pk',1))=1\right]$$

So we have

$$\begin{aligned}
&\left|\Pr_{(x',w')\leftarrow\mathcal{S}(1^\lambda)}\left[\mathcal{D}(x';1^\lambda)=1\right]-\Pr_{x'\leftarrow\mathcal{S}'(1^\lambda)}\left[\mathcal{D}(x';1^\lambda)=1\right]\right| \\
=&\left|\Pr_{\substack{(pk',sk')\leftarrow Gen(1^\lambda)\\Enc,\mathcal{A}}}\left[\mathcal{A}(pk',Enc(pk',1))=1\right]\pm negl(\lambda)-\Pr_{\substack{(pk,sk)\leftarrow Gen(1^\lambda)\\Enc,\mathcal{A}}}\left[\mathcal{A}(pk,Enc(pk,0))=1\right]\right| \\
\geq&\left|\Pr_{\substack{(pk,sk)\leftarrow Gen(1^\lambda)\\Enc,\mathcal{A}}}\left[\mathcal{A}(pk,Enc(pk,1))=1\right]-\Pr_{\substack{(pk,sk)\leftarrow Gen(1^\lambda)\\Enc,\mathcal{A}}}\left[\mathcal{A}(pk,Enc(pk,0))=1\right]\right|-negl(\lambda) \\
\geq&\, Adv^{\text{CPA}}_{\mathcal{A},\Pi_1}(\lambda)-negl(\lambda)>\frac{1}{poly(\lambda)}-negl(\lambda)
\end{aligned}$$

This contradicts the assumption that L is an average hard NP language. The theorem thus follows. ∎

3.3 PKE Scheme II

In this subsection, we will present another scheme which is slightly different from Π_1.

3.3.1 Construction of PKE Scheme

Let L be an average hard NP language with NP relation R_L and sampling algorithm $\mathcal{S}$ on L. Construct PKE scheme $\Pi_2=(Gen,Enc,Dec)$ as follows:

Construction 2: PKE scheme $\Pi_2 = (Gen, Enc, Dec)$

- $Gen(1^\lambda)$: $(x, w) \leftarrow \mathcal{S}(1^\lambda)$, set $pk = (x, \lambda)$ and $sk = w$. $|x| = \ell(\lambda), |w| = p(\lambda)$.
- $Enc(pk, b)$, $b \in \{0,1\}$:
  - Randomly select $\alpha \in \{0,1\}^\ell$ by selecting $\overline{x} \in \overline{L}$ (i.e. $\overline{x} \leftarrow \mathcal{S}(1^\lambda)$) and setting $\alpha = x \oplus \overline{x}$. Randomly select $\beta = \beta_1 \cdots \beta_p \in \{0,1\}^p$.
  - Run $Randomize$ to get circuit C: $C \leftarrow Randomize(R_L, \alpha, \beta; 1^\lambda)$, where $C : \{0,1\}^\ell \times \{0,1\}^p \to \{0,1\}$. Assume C contains t logical gate vertices. Set $s = \ell + p$.
  - If $b = 1$, set $\rho_1 = \beta$, otherwise, randomly select $\rho_0 \in \{0,1\}^p$ satisfying $\rho_0 \neq \beta$.
  - Set $\widetilde{u} = x \oplus \alpha$, and garble C with constraining $u = \widetilde{u}$, i.e.

$$\widehat{C}_b = \left\{ K^{\widetilde{u}}, \widehat{E}_b, \left\{ \widehat{V}_k \right\}_{k=s+1}^{s+t+1}, D \right\} \leftarrow Garble\left(\widetilde{u}, C, \rho_b, 1^\lambda\right)$$

    where

$$K^{\widetilde{u}} = \{K_i^{\widetilde{u}_i}\}_{i=1}^{\ell}$$

$$\widehat{E}_b = \left\{ \left(\widehat{K}_{\ell+j}^0, \widehat{K}_{\ell+j}^1 \right) \right\}_{j=1}^{p} = \left\{ \left(K_{\ell+j}^{\rho_{b,j}}, K_{\ell+j}^{1-\rho_{b,j}} \right) \right\}_{j=1}^{p}$$

$$D = \left\{ \left(0, K_{s+t+1}^0\right), \left(1, K_{s+t+1}^1\right) \right\}$$

  - Output ciphertext $c_b = \widehat{C}_b$.
- $Dec(sk, c_b)$: Interpret $c_b = \widehat{C}_b = \left\{ K^{\widetilde{u}}, \widehat{E}_b, \left\{ \widehat{V}_k \right\}_{k=s+1}^{s+t+1}, D \right\}$ as a garbled circuit. Compute $a = Eval(\widehat{C}_b, sk)$ and output a.

Correctness. It is easy to see that, compared with Π_1, Π_2 only changes the encryption of 0. In Π_2, different encoding tables are used for encrypting 0 and 1, specifically, the encoding tables for encrypting b is $\widehat{E}_b = \left\{ \left(K_{\ell+j}^{\rho_{b,j}}, K_{\ell+j}^{1-\rho_{b,j}} \right) \right\}_{j=1}^{p}$. This results in an negligible error when decrypting the ciphertext of 0. Specifically, the following claim holds:

Claim 4 Assume $(x, w) \leftarrow \mathcal{S}(1^\lambda)$, $pk = (x, \lambda)$ and $sk = w$. We have that

$$(1) \Pr_{\substack{(pk,sk) \leftarrow \mathcal{S}(1^\lambda) \\ Enc}} [Dec(sk, Enc(pk, 1)) = 1] = 1$$

$$(2) \Pr_{\substack{(pk,sk) \leftarrow \mathcal{S}(1^\lambda) \\ Enc}} [Dec(sk, Enc(pk, 0)) = 0] \geq 1 - negl(\lambda)$$

Proof. From the definitions of $Garble$ and $Randomize$, it holds that

$$\begin{aligned} \widehat{C}_b(v) &= Eval(\widehat{C}_b, v) \\ &= C(\widetilde{u}, v \oplus \rho_b) \\ &= R_\Gamma(\pi_0(F(\widetilde{u})), \pi_1(F(\widetilde{u}), \widetilde{u}, v \oplus \rho_b)) \end{aligned} \tag{19}$$

where for $\widetilde{u} = x \oplus \alpha$, F is a one-to-one function, $\pi = (\pi_0, \pi_1, \pi_1')$ is Levin reduction from L' with NP relation $R_{L'}$ to Γ with NP relation R_Γ.

(1) Recall that L' is defined as follows:

$$L' = \{y : \exists u, v, \text{ such that } y = F(u), R_L(u \oplus \alpha, v \oplus \beta) = 1\}$$

Clearly, $R_L(x, w) = 1$ means that $R_{L'}(F(\widetilde{u}), (\widetilde{u}, w \oplus \rho_1)) = 1$, which further means that $R_\Gamma(\pi_0(F(\widetilde{u})), \pi_1(F(\widetilde{u}), \widetilde{u}, w \oplus \rho_1)) = 1$ because π is Levin reduction from L' to Γ. Therefore, it follows that $\widehat{C}_1(sk) = 1$. In other words, we have that

$$\Pr_{\substack{(pk,sk)\leftarrow\mathcal{S}(1^\lambda)\\ Enc}}[Dec(sk, Enc(pk, 1)) = 1] = \Pr_{\substack{(pk,sk)\leftarrow\mathcal{S}(1^\lambda)\\ Enc}}[Eval(\widehat{C}_1, sk) = 1] = 1$$

(2) Let

$$\varepsilon = \Pr_{\substack{(pk,sk)\leftarrow\mathcal{S}(1^\lambda)\\ Enc}}[Dec(sk, Enc(pk, 0)) = 1] \tag{20}$$

Form Equation (19) and the definition of Dec, we have that

$$\begin{aligned}\Pr_{\substack{(pk,sk)\leftarrow\mathcal{S}(1^\lambda)\\ Enc}}[Dec(sk, Enc(pk, 0)) = 1] &= \Pr_{\substack{(pk,sk)\leftarrow\mathcal{S}(1^\lambda)\\ Enc}}[\widehat{C}_0(sk) = 1] \\ &= \Pr_{\substack{(pk,sk)\leftarrow\mathcal{S}(1^\lambda)\\ Enc}}[R_\Gamma(\pi_0(\widetilde{y}), \pi_1(\widetilde{y}, \widetilde{u}, w')) = 1] \end{aligned} \tag{21}$$

where $\widetilde{y} = F(\widetilde{u})$ and $w' = sk \oplus \rho_0$. Since ρ_0 is randomly selected, w' is uniformly distributed. Note that $R_\Gamma(\pi_0(\widetilde{y}), \pi_1(\widetilde{y}, \widetilde{u}, w')) = 1$ means that $\pi_1(\widetilde{y}, \widetilde{u}, w')$ is a witness of $\pi_0(\widetilde{y}) \in \Gamma$. Following the fact that $\pi = (\pi_0, (\pi_1, \pi_1'))$ is Levin reduction from L' to Γ, we have that $\pi_1'(\widetilde{y}, \pi_1(\widetilde{y}, \widetilde{u}, w'))$ is a witness of $\widetilde{y} \in L'$. Therefore, it follows from Equation (20) and (21) that

$$\Pr_{\substack{(pk,sk)\leftarrow\mathcal{S}(1^\lambda)\\ Enc}}[R_{L'}(\widetilde{y}, \pi_1'(\widetilde{y}, \pi_1(\widetilde{y}, \widetilde{u}, w')) = 1] \geq \varepsilon \tag{22}$$

Let $(u', v') = \pi_1'(\widetilde{y}, \pi_1(\widetilde{y}, \widetilde{u}, w'))$. From $R_{L'}(\widetilde{y}, (u', v')) = 1$, we know that (u', v') is a witness for $\widetilde{y} \in L'$. By the definition, it must holds that $\widetilde{y} = F(u')$ and $v' \oplus \beta$ is a witness of $u' \oplus \alpha \in L$. Recall that F is one-to-one, so we can get $u' = \widetilde{u}$, i.e. $u' \oplus \alpha = \widetilde{u} \oplus \alpha = x$, from $\widetilde{y} = F(\widetilde{u}) = F(u')$. Therefore, $v' \oplus \beta$ is a witness of $\widetilde{u} \oplus \alpha = x \in L$. If ε is non-negligible, then Equation (22) (i.e. Equation (20)) means that there exists PPT algorithm $\mathcal{B}$ breaking the assumption that L is average hard. The details of $\mathcal{B}$ are as follows:

$\mathcal{B}(x; 1^\lambda)$

On inputting x ($|x| = \ell$) and security parameter λ, $\mathcal{B}(x; 1^\lambda)$ do:

- Randomly select $\alpha \in \{0,1\}^\ell$ by selecting $\overline{x} \in \overline{L}$ (i.e. $\overline{x} \leftarrow \mathcal{S}(1^\lambda)$) and setting $\alpha = x \oplus \overline{x}$. Randomly select $\beta = \beta_1 \cdots \beta_p \in \{0,1\}^p$.
- Run $Randomize(R_L, \alpha, \beta; 1^\lambda)$, record F (one-to-one function) and $\pi = (\pi_0, \pi_1, \pi_1')$ (Levin reduction from L' to Γ) and finally obtain C:

$$C \leftarrow Randomize(R_L, \alpha, \beta; 1^\lambda)$$

- Compute $\widetilde{u} = x \oplus \alpha$ and $\widetilde{y} = F(\widetilde{u})$.
- Randomly select $w' \in \{0,1\}^p$, compute $z' = \pi_1(\widetilde{y}, \widetilde{u}, w')$.
- If $R_\Gamma(\pi_0(\widetilde{y}), z') = 1$, compute $(u', v') = \pi_1'(\widetilde{y}, z') \in \{0,1\}^{\ell+p}$.
- Output $v' \oplus \beta$.

Clearly, $R_\Gamma(\pi_0(\widetilde{y}), z') = 1$ (that is, z' is the witness for $\pi_0(\widetilde{y}) \in \Gamma$) means that $(u', v') = \pi'_1(\widetilde{y}, z') \in \{0,1\}^{\ell+p}$ is a witness for $\widetilde{y} \in L'$. That is, (u', v') satisfies $\widetilde{y} = F(u')$ and $R_L(u' \oplus \alpha, v' \oplus \beta) = 1$. Since $\widetilde{y} = F(\widetilde{u}) = F(u')$, we have that $u' = \widetilde{u}$ and $R_L(x, v' \oplus \beta) = R_L(u' \oplus \alpha, v' \oplus \beta) = 1$. Therefore, we have that

$$\Pr_{\mathcal{B}}[R_L(x,w) = 1 : w \leftarrow \mathcal{B}(x, 1^\lambda)] = \Pr_{\mathcal{B}}[R_\Gamma(\pi_0(\widetilde{y}), \pi_1(\widetilde{y}, \widetilde{u}, w')) = 1] \geq \varepsilon \tag{23}$$

If ε is non-negligible, Equation (23) contradicts the assumption that L is average hard. In other words, if L is average hard, it must hold that

$$\Pr_{\substack{(pk,sk)\leftarrow(1^\lambda)\\ Enc}}[Dec(sk, Enc(pk, 0)) = 0] \geq 1 - negl(\lambda)$$

∎

3.3.2 Proof of Security

We first prove the following lemma.

Lemma 4. *For $x \leftarrow \mathcal{S}'(1^\lambda)$, then $\widehat{C}_1 \leftarrow Enc(pk, 1)$ and $\widehat{C}_0 \leftarrow Enc(pk, 0)$ are computationally indistinguishable, where $pk = (x, \lambda)$. That is, for any PPT algorithm $\mathcal{A}$, it holds that*

$$\left| \Pr_{\substack{x\leftarrow\mathcal{S}'(1^\lambda)\\ \mathcal{A},Enc}} \left[\mathcal{A}((x,\lambda), \widehat{C}_0) = 1\right] - \Pr_{\substack{x\leftarrow\mathcal{S}'(1^\lambda)\\ \mathcal{A},Enc}} \left[\mathcal{A}((x,\lambda), \widehat{C}_1) = 1\right] \right| = negl(\lambda) \tag{24}$$

Proof. Let $C \leftarrow Randomize(R_L, \alpha, \beta; 1^\lambda)$, where α, β are randomly selected. By the definition of $Enc(pk, b)$, we have

$$\widehat{C}_b = \left\{ K^{\widetilde{u}}, \widehat{E}_b, \left\{\widehat{V}_k\right\}_{k=s+1}^{s+t}, D \right\} \leftarrow Garble\left(\widetilde{u}, C, \rho_b; 1^\lambda\right)$$

where $\widetilde{u} = x \oplus \alpha$, $\rho_b = \rho_{b,1} \cdots \rho_{b,p} \in \{0,1\}^p$ is randomly selected and satisfies $\rho_1 = \beta$ and $\rho_0 \neq \beta$, C compute

$$\Psi(u, v) = R_\Gamma(\pi_0(F(u)), \pi_1(F(u), (u, v)))$$

where $\pi = (\pi_0, \pi_1, \pi'_1)$ is Levin reduction from L':

$$L' = \{y : \exists u, v, \text{ such that } y = F(u), R_L(u \oplus \alpha, v \oplus \beta) = 1\}$$

to Γ with NP relation R_Γ.

To prove $((x,\lambda), \widehat{C}_0)$ and $((x,\lambda), \widehat{C}_1)$ are computationally indistinguishable, we select $\rho' \in \{0,1\}^p$ randomly , set $\widetilde{u}' = x$ and let

$$\widehat{C}' \leftarrow Garble\left(\widetilde{u}', C, \rho'; 1^\lambda\right)$$

and then prove that both $(x, \widehat{C}_0)$ and $(x, \widehat{C}_1)$ are computationally indistinguishable from $(x, \widehat{C}')$. This means that $((x,\lambda), \widehat{C}_0)$ and $((x,\lambda), \widehat{C}_1)$ are computationally indistinguishable.

Since $x, \overline{x} \notin L$, we have that $F(\widetilde{u}) = F(x \oplus \alpha) \notin L'$ and $F(\widetilde{u}') = F(\overline{x} \oplus \alpha) \notin L'$. It follows that $C(\widetilde{u}, v) = C(\widetilde{u}', v) \equiv 0$ for any $v \in \{0,1\}^p$.

1) $(x, \widehat{C}')$ and $(x, \widehat{C}_0)$ are computationally indistinguishable. In fact, by Claim 3 (given in Subsection 2.4), we have that $((\widetilde{u}', \rho'), (\widetilde{u}, \rho_0), \widehat{C}')$ and $((\widetilde{u}', \rho'), (\widetilde{u}, \rho_0), \widehat{C}_0)$ are computationally indistinguishable. Therefore, $(x, \widehat{C}') = (\widetilde{u}', \widehat{C}')$ and $(x, \widehat{C}_0) = (\widetilde{u}', \widehat{C}_0)$ are computationally indistinguishable.
2) $(x, \widehat{C}')$ and $(x, \widehat{C}_1)$ are computationally indistinguishable. Similar to 1), it follows from Claim 3 that $((\widetilde{u}', \rho'), (\widetilde{u}, \rho_1), \widehat{C}')$ and $((\widetilde{u}', \rho'), (\widetilde{u}, \rho_1), \widehat{C}_1)$ are computationally indistinguishable. Therefore, $(x, \widehat{C}') = (\widetilde{u}', \widehat{C}')$ and $(x, \widehat{C}_1) = (\widetilde{u}', \widehat{C}_1)$ are computationally indistinguishable.

Combining (1) and (2), we have that $(x, \widehat{C}_0)$ and $(x, \widehat{C}_1)$ are computationally indistinguishable. It follows that $((x, \lambda), \widehat{C}_0)$ and $((x, \lambda), C_1)$ are computationally indistinguishable. That is, Equation (24) holds. This concludes the proof. ∎

Theorem 2. *The PKE scheme Π_2 is CPA secure when L is an average hard NP language.*

Proof. If Π_2 is not CPA secure, then there exist an algorithm $\mathcal{D}$ to attack the average hardness of L.

Suppose towards contradiction that there exists a PPT adversary $\mathcal{A}$ breaking Π_2's CPA security with non-negligible advantage, that is, there exists a polynomial $poly$, such that

$$Adv^{\text{CPA}}_{\mathcal{A},\Pi_2}(\lambda)=\left|\Pr_{\substack{(x,w)\leftarrow\mathcal{S}(1^\lambda)\\ \mathcal{A},Enc}}\left[\mathcal{A}(pk,\widehat{C}_1)=1\right]-\Pr_{\substack{(x,w)\leftarrow\mathcal{S}(1^\lambda)\\ \mathcal{A},Enc}}\left[\mathcal{A}(pk,\widehat{C}_0)=1\right]\right|\geq\frac{1}{poly(\lambda)} \quad (25)$$

holds for infinitely λ's. To obtain a contradiction we construct algorithm $\mathcal{D}$ to break the hardness of L. The details of $\mathcal{D}$ are as follows:

$\mathcal{D}(x; 1^\lambda)$

On inputting x and security λ, $\mathcal{D}(x; 1^\lambda)$ do:
- Set $pk = (x, \lambda)$ and randomly select $b \in \{0, 1\}$.
- $\widehat{C}_b \leftarrow Enc(pk, b)$.
- Invoke $\mathcal{A}$ with $(pk, \widehat{C}_b)$ and obtain $\mathcal{A}$'s return b': $b' \leftarrow \mathcal{A}(1^\lambda, pk, \widehat{C}_b)$.
- If $b = b'$, then $\sigma = 1$, otherwise, $\sigma = 0$.
- Output σ.

From the strategy of $\mathcal{D}$, it is easy to obtain that

$$\begin{aligned}
&\Pr_{\substack{(x,w)\leftarrow\mathcal{S}(1^\lambda)\\ \mathcal{D}}}\left[\mathcal{D}(x;1^\lambda)=1\right]\\
=&\Pr_{\substack{(x,w)\leftarrow\mathcal{S}(1^\lambda)\\ \mathcal{A},Enc}}\left[\mathcal{A}(1^\lambda,pk,\widehat{C}_b)=b\right]\\
=&\frac{1}{2}\Pr_{\substack{(x,w)\leftarrow\mathcal{S}(1^\lambda)\\ \mathcal{A},Enc}}\left[\mathcal{A}(pk,\widehat{C}_1)=1\right]+\frac{1}{2}\Pr_{\substack{(x,w)\leftarrow\mathcal{S}(1^\lambda)\\ \mathcal{A},Enc}}\left[\mathcal{A}(pk,\widehat{C}_0)=0\right]\\
=&\frac{1}{2}+\frac{1}{2}\left(\Pr_{\substack{(x,w)\leftarrow\mathcal{S}(1^\lambda)\\ \mathcal{A},Enc}}\left[\mathcal{A}(pk,\widehat{C}_1)=1\right]-\Pr_{\substack{(x,w)\leftarrow\mathcal{S}(1^\lambda)\\ \mathcal{A},Enc}}\left[\mathcal{A}(pk,\widehat{C}_0)=1\right]\right) \quad (26)
\end{aligned}$$

and

$$\begin{aligned}
&\Pr_{\substack{x\leftarrow\mathcal{S}'(1^\lambda)\\ \mathcal{D}}}\left[\mathcal{D}(x;1^\lambda)=1\right]\\
=&\Pr_{\substack{x\leftarrow\mathcal{S}'(1^\lambda)\\ \mathcal{A},Enc}}\left[\mathcal{A}(pk,\widehat{C}_b)=b\right]\\
=&\frac{1}{2}\Pr_{\substack{x\leftarrow\mathcal{S}'(1^\lambda)\\ \mathcal{A},Enc}}\left[\mathcal{A}(pk,\widehat{C}_1)=1\right]+\frac{1}{2}\Pr_{\substack{x\leftarrow\mathcal{S}'(1^\lambda)\\ \mathcal{A},Enc}}\left[\mathcal{A}(pk,\widehat{C}_0)=0\right]\\
=&\frac{1}{2}+\frac{1}{2}\left(\Pr_{\substack{x\leftarrow\mathcal{S}'(1^\lambda)\\ \mathcal{A},Enc}}\left[\mathcal{A}(pk,\widehat{C}_1)=1\right]-\Pr_{\substack{x\leftarrow\mathcal{S}'(1^\lambda)\\ \mathcal{A},Enc}}\left[\mathcal{A}(pk,\widehat{C}_0)=1\right]\right) \qquad (27)
\end{aligned}$$

Therefore, combining Equation (26) with (27) we have that

$$\begin{aligned}
&\left|\Pr_{\substack{(x,w)\leftarrow\mathcal{S}(1^\lambda)\\ \mathcal{D}}}\left[\mathcal{D}(x;1^\lambda)=1\right]-\Pr_{\substack{x\leftarrow\mathcal{S}'(1^\lambda)\\ \mathcal{D}}}\left[\mathcal{D}(x;1^\lambda)=1\right]\right|\\
\geq&\left|\frac{1}{2}\left(\Pr_{\substack{(x,w)\leftarrow\mathcal{S}(1^\lambda)\\ \mathcal{A},Enc}}\left[\mathcal{A}(pk,\widehat{C}_1)=1\right]-\Pr_{\substack{(x,w)\leftarrow\mathcal{S}(1^\lambda)\\ \mathcal{A},Enc}}\left[\mathcal{A}(pk,\widehat{C}_0)=1\right]\right)\right|\\
&-\left|\frac{1}{2}\left(\Pr_{\substack{x\leftarrow\mathcal{S}'(1^\lambda)\\ \mathcal{A},Enc}}\left[\mathcal{A}(pk,\widehat{C}_1)=1\right]-\Pr_{\substack{x\leftarrow\mathcal{S}'(1^\lambda)\\ \mathcal{A},Enc}}\left[\mathcal{A}(pk,\widehat{C}_0)=1\right]\right)\right|
\end{aligned}$$

From Lemma 4 and the assumption (Equation (25)), we further obtain

$$\left|\Pr_{\substack{(x,w)\leftarrow\mathcal{S}(1^\lambda)\\ \mathcal{D}}}\left[\mathcal{D}(x;1^\lambda)=1\right]-\Pr_{\substack{x\leftarrow\mathcal{S}'(1^\lambda)\\ \mathcal{D}}}\left[\mathcal{D}(x;1^\lambda)=1\right]\right|\geq\frac{1}{2}\left(\frac{1}{poly(\lambda)}-negl(\lambda)\right)$$

This contradicts the assumption that L is an average hard NP language. The theorem thus follows. ∎

4 Witness Encryption Scheme

In this section, we will show that the construction of PKE scheme Π_1 in fact presents a WE scheme. By the definition of Π_1, Enc take as inputs $pk=(x,\lambda)$ (an instance x and security parameter λ) and $b\in\{0,1\}$ to produce a garbled circuit $\widehat{C}_b$ as the ciphertext of b, and Dec use w to decrypt $\widehat{C}_b$, satisfying $\Pr[Dec(Enc(x,b,1^\lambda),w)=b]=1-negl(\lambda)$ when $R_L(x,w)=1$. Witness encryption derived from $\Pi_1=(Gen,Enc,Dec)$ is as follows:

Construction 3: Witness Encryption $\Pi = (Encrypt, Decrypt)$

Assume $L \in NP$. Enc, Dec are defined as in Construction 1.

- $Encrypt(x, b, 1^\lambda)$, $b \in \{0,1\}$, $|x| = \ell$:
  - Set $pk = (x, \lambda)$, and run $Enc(pk, b)$: $\widehat{C}_b \leftarrow Enc(pk, b)$,
  - Output ciphertext $\widehat{C}_b$.
- $Decrypt(w, c_b)$: Interpret $\widehat{C}_b = \left\{ K^{\widetilde{u}}, \widehat{E}, \left\{ \widehat{V}_k \right\}_{k=s+1}^{s+t+1}, D \right\}$ as a garbled circuit.

  Compute $a = Eval(\widehat{C}_b, w)$ and output a.

Correctness. Follows from the correctness of Π_1.

Soundness. Soundness security requires that $Encrypt(1^\lambda, x, 0)$ and $Encrypt(1^\lambda, x, 1)$ are indistinguishable when $x \notin L$, that is, for any $x \notin L$, any PPT adversary $\mathcal{A}$, it holds that

$$\left|\Pr[\mathcal{A}(Encrypt(1^\lambda, x, 0)) = 1] - \Pr[\mathcal{A}(Encrypt(1^\lambda, x, 1)) = 1]\right| < negl(\lambda) \tag{28}$$

By the definition, we have that

$$\widehat{C}_0 \leftarrow Encrypt(x, 0, 1^\lambda) = Garble(u_0, C, \beta; 1^\lambda), u_0 = x' \oplus \alpha \tag{29}$$

$$\widehat{C}_1 \leftarrow Encrypt(x, 1, 1^\lambda) = Garble(u_1, C, \beta; 1^\lambda), u_1 = x \oplus \alpha \tag{30}$$

where $\alpha \in \{0,1\}^\ell, \beta \in \{0,1\}^p$ are randomly selected, $x' \notin L, C \leftarrow Randomize(R_L, \alpha, \beta; 1^\lambda)$. Since $x, x' \notin L$, it follows from the definition of $Randomize$ that $C(u_1, v) = C(u_0, v) \equiv 0$ for any v. Note that $\widehat{C}_0$ and $\widehat{C}_1$ differ only because of $u_1 \neq u_0$. By Claim 1 (given in Subsection 2.4), we have that $(u_0, u_1, \widehat{C}_0)$ and $(u_0, u_1, \widehat{C}_1)$ are computationally indistinguishable, and thus $\widehat{C}_0$ and $\widehat{C}_1$ are computationally indistinguishable. That is, Equation (28) holds.

On the other hands, comparing Equation (29) and (30), we can find that $\widehat{C}_0 \leftarrow Encrypt(x, 0, 1^\lambda)$ is the same as $\widehat{C}'_1 \leftarrow Encrypt(x', 1, 1^\lambda)$, while $\widehat{C}_1 \leftarrow Encrypt(x, 1, 1^\lambda)$ can also be interpreted as $\widehat{C}'_0 \leftarrow Encrypt(x', 0, 1^\lambda)$, that is,

$$Encrypt(x, 0, 1^\lambda) = Encrypt(x', 1, 1^\lambda)$$

$$Encrypt(x, 1, 1^\lambda) = Encrypt(x', 0, 1^\lambda)$$

This means that for a given ciphertext $\widehat{C}^* \leftarrow Encrypt(x, b, 1^\lambda)$ (where $x \notin L$), there exists $x' \notin L$ such that $\widehat{C}^*$ can be interpreted as $\widehat{C}^* \leftarrow Encrypt(x', 1-b, 1^\lambda)$. In other words, a ciphertext of b relative to x is a ciphertext of $1-b$ relative to x'. Therefore, it is completely impossible for an adversary to extract b from a given ciphertext $\widehat{C}_b \leftarrow Encrypt(x, b, 1^\lambda)$ without auxiliary input x. This also means that for any PPT adversary $\mathcal{A}$, Equation (28) holds.

Therefore, we conclude the following conclusion:

Theorem 3. *Let $L \in NP$. If Enc, Dec are the same as in Construction 1, then $\Pi = (Encrypt, Decrypt)$ given by Construction 3 is a bit WE scheme for L.*

The soundness of WE requires an adversary to distinguish pairs of ciphertexts relative to instances $x \notin L$ without obtaining x. In fact, Π in fact satisfies a stronger soundness, which allows adversary to obtain the auxiliary input x. That is, for any PPT adversary $\mathcal{A}$, it holds that

$$\left|\Pr[\mathcal{A}(x, Encrypt(1^\lambda, x, 0)) = 1] - \Pr[\mathcal{A}(x, Encrypt(1^\lambda, x, 1)) = 1]\right| < negl(\lambda) \tag{31}$$

The proof of this is simple. Recall $\alpha = x \oplus \overline{x}$, where $\overline{x} \notin L$. Let $u' = x$ and $\widehat{C}' \leftarrow Garble\left(u', C, \beta; 1^{\lambda}\right)$, It is easy to see that $C(u', v) = C(u_0, v) \equiv 0$ for any $v \in \{0,1\}^p$. Therefore, by Claim 1 (given in Subsection 2.4), we have that $(u', u_0, \widehat{C}_0)$ and $(u', u_0, \widehat{C}')$ are computationally indistinguishable. It follows that $(x, \widehat{C}_0) = (u', \widehat{C}_0)$ and $(x, \widehat{C}') = (u', \widehat{C}')$ are computationally indistinguishable. Similarly, we can obtain that $(x, \widehat{C}_1) = (u', \widehat{C}_1)$ and $(x, \widehat{C}') = (u', \widehat{C}')$ are computationally indistinguishable. And thus, we have that $(x, \widehat{C}_0)$ and $(x, \widehat{C}_1)$ are computationally indistinguishable. That is, Equation (31) holds.

Soundness of WE is only for $x \notin L$ and so it is different from CPA security of PKE, which is for normal public-key. Specifically, Π_1's CPA security is based on the assumption that $L \in NP$ is average hard, while Π's soundness is unconditional and has nothing to do with whether L is average hard or not. The following corollary is immediately apparent.

Corollary. For any $L \in NP$, if there exist sampling algorithms $\mathcal{S}$ on L and $\mathcal{S}'$ on $\overline{L}$, then there exists WE scheme for L.

Note that Garg et al. in [13] showed that it is impossible to reduce breaking the soundness of WE scheme to solving the simple non-interactive assumption on which the soundness is based. Obviously, this impossibility does not seem to contradict our results, because it does not rule out the possibility that there is a WE scheme that does not require any difficulty assumption.

In addition, Garg et al. in [13] presented ways to obtain non-black-box constructions of PKE and identity-based encryption (IBE) and attribute-based encryption (ABE) from WE and OWF. Therefore, by our WE scheme $\Pi = (Encrypt, Decrypt)$ and the constructions of [13], we can not only obtain non-black-box construction of PKE from OWF, but also non-black-box construction of IBE and ABE from OWF.

References

1. 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, pages 1-18, 2001.
2. Boaz Barak. How to go beyond the black-box simulation barrier. In Proceedings of FOCS '01, pages 106-115, 2001.
3. Boaz Barak. The complexity of public-key cryptography. In: Lindell, Y. (ed.) Tutorials on the Foundations of Cryptography, pp. 45-77. Springer International Publishing (2017).
4. Itay Berman, Akshay Degwekar, Ron D. Rothblum, Prashant Nalini Vasudevan. From Laconic Zero-Knowledge to Public-Key Cryptography. Electron. Colloquium Comput. Complex. TR17 (2017).
5. Mihir Bellare, Viet Tung Hoang, Phillip Rogaway. Foundations of garbled circuits. In Proceedings of the 2012 ACM conference on Computer and communications security, 2012, pages 784-796 (2012).
6. Zvika Brakerski, Jonathan Katz and Gil Segev, Arkady Yerukhimovich. Limits on the power of zero-knowledge proofs in cryptographic constructions. In Theory of Cryptography Conference, 2011, pp 559-578.
7. Boaz Barak, Yehuda Lindell. Strict polynomial-time in simulation and extraction. SIAM Journal on Computing, 33(4):783¨C818, August 2004.
8. Yilei Chen, Vinod Vaikuntanathan, Hoeteck Wee. GGH15 beyond permutation branching programs: Proofs, attacks, and candidates. In Advances in Cryptology ¨C CRYPTO 2018, pages 577-607.
9. Dana Dachman-Soled. Towards Non-Black-Box Separations of Public Key Encryption and One Way Function. In Theory of Cryptography Conference. Springer, Berlin, Heidelberg, LNCS 9986, pages 169-191,2016.

10. Whitfield Diffie, Martin E. Hellman. New directions in cryptography. IEEE Trans. Inf. Theory 22(6), 644¨C654 (1976), https://doi.org/10.1109/TIT.1976.1055638.
11. Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, Brent Waters. Candidate Indistinguishability Obfuscation and Functional Encryption for all Circuits. In IEEE 54th Annual Symposium on Foundations of Computer Science (2013), pages 40-49.
12. Oded Goldreich, Shafi Goldwasser, Silvio Micali. How to construct random functions. Journal of the Acm, 1986, 33(4):792-807.
13. Sanjam Garg, Craig Gentry, Amit Sahai, Brent Waters. Witness Encryption and its Applications. In Proceedings of the forty-fifth annual ACM symposium on Theory of computing. pages 467-476, 2013.
14. Sanjam Garge, Mohammad Hajiabadi, Mohammad Mahmoody, Ameer Mohammed. Limits on the Power of Garbling Techniques for Public-Key Encryption. In Advances in Cryptology-CRYPTO(2018), pages 335-364,2018.
15. Craig Gentry, Allison B. Lewko, Brent Waters. Witness Encryption from Instance Independent Assumptions. In Advances in cryptology - CRYPTO 2014, pages 426-443.
16. Russell Impagliazzo, Michael Luby. One-way functions are essential for complexity based cryptography. In IEEE 30th Annual Symposium on Foundations of Computer Science, 1989.
17. Russell. Impagliazzo. A personal view of average-case complexity. In Proceedings of the Tenth Annual Structure in Complexity Theory Conference, Minneapolis, Minnesota, USA, June 19-22, 1995. pp. 134-147. IEEE Computer Society (1995), https://doi.org/10.1109/SCT.1995.514853.
18. Russell Impagliazzo, Steven Rudich. Limits on the the provable consequences of one-way permutations. In CRYPTO '88: Proceedings on Advances in cryptology, February 1990, Pages 8¨C26.
19. Mohammad Mahmoody, Rafael Pass. The Curious Case of Non-Interactive Commitments ¨C On the Power of Black-Box vs. Non-Black-Box Use of Primitives. In Advances in Cryptology-CRYPTO, 2012.
20. Minh-Huyen Nguyen, Shien Jin Ong, Salil P. Vadhan. Statistical Zero-Knowledge Arguments for NP from Any One-Way Function. In 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings. IEEE, 2006.
21. Omkant Pandey, Manoj Prabhakaran, Amit Sahai. Obfuscation-based Non-black-box Simulation and Four Message Concurrent Zero Knowledge for NP. In Theory of Cryptography Conference, 2015.
22. Omer Reingold, Luca Trevisan, Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In TCC, pages 1-20, 2004.
23. Andrew Chi-Chih. Yao. Protocols for secure computations. In 23rd annual symposium on foundations of computer science (SFCS 1982). IEEE, 1982, pages 160-164.

A Yao's garbled circuit scheme

Garbled circuit was first presented by Yao[23], and has been formalized by Bellare et al. [5]. According to the formal language of [5], a garbling circuit scheme $Garble$ is defined by a tuple algorithms, $Garble = (Gb, En, Ev, De, Ve)$.

Definition 4. *(Garbled Circuits) A garbling circuit scheme consists of a tuple of polynomial algorithms* $Garble = (Gb, En, Ev, De, Ve)$.

- $Gb(1^n, C)$*, taking the security parameter* n *and a circuit* $C : \{0,1\}^\ell \to \{0,1\}^k$ *as input, outputs a garbled circuit* $\widehat{C}$ *of* C *and a pair of keys* (e, d)*, where* e *is an encoding key and* d *is decoding list.*
- $En(e, x)$*, taking* $x \in \{0,1\}^\ell$ *and encoding key* e *as input, outputs the garbled encoding* $\widehat{x}$ *of* $x \in \{0,1\}^\ell$.

- $Ev(\widehat{C}, \widehat{x})$ *evaluates garbled circuit* $\widehat{C}$ *on garbled encoding* $\widehat{x}$*, and outputs a garbled output* $\widehat{y}$*.*
- $De(d, \widehat{y})$ *outputs the decoding of* $\widehat{y}$*.*
- $Ve(C, \widehat{C}, e)$ *output 1 if* $\widehat{C}$ *is a valid garbling circuit of* C*, output 0 otherwise.*

A garbling scheme is called correct if for any x, there is $De(d, Ev(\widehat{C}, En(e, x))) = C(x)$. Garbled circuit is used for secure two-party computation, in where both parties wish to carry out a joint computation of a shared function without disclosing their private input. So, it needs to satisfy privacy and authenticity, roughly speaking, it require $(\widehat{C}, En(e, x), d)$ does not leak any information about x except for $C(x)$ The details are not given here, see [5].

Yao's garbled circuits scheme [23] is as follws. Let $C : \{0,1\}^{\ell} \to \{0,1\}^{k}$ be a acyclic circuit that has $m + k$ gates. So C has $t = \ell + m + k$ wires, denoted as $w_1, \cdots, w_t$, where $w_1, \cdots, w_\ell$ are the input wires and $w_{t-k+1}, \cdots, w_t$ are the output wires of C. To garble C, Gb first selects a pair of key (K_i^0, K_i^1) from key space $\mathcal{K}$ to represent the bit values of 0 or 1 on wire w_i $(i = 1, \cdots, t)$, and sets $e = \{K_i^0, K_i^1\}_{i=1}^{\ell}$, $d = \{(0, K_{t-k+1}^0), (1, K_{t-k+1}^1), \cdots, (0, K_t^0), (1, K_t^1)\}$.

Once all the keys for the wires in the circuit have been chosen, garble each gate in C as follows: For any gate g with two input wires w_i, w_j and one output wire w_o, compute $c_{a,b} = E_{K_i^a}(E_{K_j^b}(K_o^{g(a,b)}))$, $a, b = 0, 1$. The garbled gate of g is represented by a "garbled computation table" $\widehat{g} = (c_0, c_1, c_2, c_3)$ which is the random order of $(c_{0,0}, c_{0,1}, c_{1,0}, c_{1,1})$. The garbled circuit $\widehat{C}$ of C consist of the garbled gate for each gate, $\widehat{C} = \{\{\widehat{g}\}_{g \in C}\}$.

For any $x = x_1 \cdots x_\ell$, $\widehat{x} = En(e, x) = \{K_1^{x_1}, \cdots, K_\ell^{x_\ell}\}$ is the encoding of x. $Ev(\widehat{C}, \widehat{x})$ computes each gate according its "garbled computation table", gate by gate, and obtains the output of k output gates, $\widehat{y} = Ev(\widehat{C}, \widehat{x})$. Finally, $De(d, \widehat{y})$ decodes $\widehat{y}$, $De(d, \widehat{y}) = z_1 \cdots z_k$ if and only if $\widehat{y} = (K_{t-k+1}^{z_1}, \cdots, K_t^{z_k})$. $Ve(C, \widehat{C}, e)$ verifies every $\widehat{g} = (c_0, c_1, c_2, c_3)$, gate by gate, by decrypting c_0, c_1, c_2, c_3. $Ve(C, \widehat{C}, e) = 1$ if all the verifications pass, $Ve(C, \widehat{C}, e) = 0$ otherwise.

It is easy to see that Yao's garbled circuits scheme is correct. In addition, Yao's garbled circuits scheme satisfies privacy and authenticity[5].