Hindawi
Security and Communication Networks
Volume 2019, Article ID 8567534, 10 pages
https://doi.org/10.1155/2019/8567534

Research Article

An Indistinguishably Secure Function Encryption Scheme

Ping Zhang, Yamin Li, and Muhua Liu

School of Mathematics and Statistics, Henan University of Science and Technology, Luoyang, China

Correspondence should be addressed to Yamin Li; 2390043823@qq.com

Received 1 April 2019; Revised 17 August 2019; Accepted 30 August 2019; Published 5 November 2019

Guest Editor: Leonel Sousa

In this work, we first design a function encryption scheme by using key encapsulation. We combine public key encryption with symmetric encryption to implement the idea of key encapsulation. In the key encapsulation, we use a key to turn a message (plaintext) into a ciphertext by symmetric encryption, and then we use public key encryption to turn this key into another ciphertext. In the design of function encryption scheme, we use the public key encryption system, symmetric encryption system, noninteractive proof system, indistinguishable obfuscator, and commitment scheme. Finally, we prove the indistinguishable security of our function encryption scheme.

1. Introduction

Nowadays, network technology has been rapidly developed and widely used. Due to the emergence of hacking, virus, and electronic fraud and theft, various information leakage incidents occur frequently. Both personal information security and enterprise information security are getting more and more attention, which makes information security more important than ever.

With the rise of cloud computing, in order to save local storage resources, more and more companies or individuals choose to store local data on third-party servers (such as cloud computing platforms). To prevent user data from leaking, the data can be encrypted before storing it on third-party servers. It is noteworthy that in this application, third-party servers need to operate something on the user's ciphertext, such as data mining, query, and statistics of the ciphertext. However, the traditional encryption system does not support ciphertext computing.

This problem can be solved by using fully homomorphic encryption [1]. It is possible to compute on the ciphertext using fully homomorphic encryption. In other words, a nontrusted information processing system can process information in ciphertext environment without disclosing any information of users. However, the main problem of fully homomorphic encryption is that the results are encrypted. That is to say, although the third-party servers can perform ciphertext calculation, it cannot obtain the calculation results. In many application scenarios, third-party servers need to make certain decisions based on the results of the calculations. Therefore, fully homomorphic encryption does not meet our requirements.

Function encryption firstly appeared in the article [2]. It is an extension of the articles [2–7]. It enables the third party to operate and output a function of plaintext without decryption. Therefore, it is very suitable for the computing scenario of data outsourcing encryption. Specifically, in function encryption for a function family, the key generation algorithm can generate the corresponding decryption key for each function in this function family by using the master private key. Given the ciphertext of a plaintext, the owner of the decryption key of each function can calculate the corresponding function value for this plaintext. A secure function encryption system should satisfy that users with decryption key can only get a function value of the plaintext but not any information about the plaintext. The traditional public key encryption scheme is too extreme. It enables users with decrypted key to get all or no messages from the ciphertext. Therefore, it is of great significance to study function encryption.

Function encryption is a perfect noninteractive solution to many problems that arise in delegating services to external servers. Consider the following scenario: Suppose a bank subscribes to an external cloud server for storing financial

records of its customers. In order to ensure the security of these records and perform various computations on the outsourced data remotely from time to time, a cost-effective choice for the bank is to use a function encryption scheme to encrypt the records locally prior to uploading to the cloud server. Now, suppose the researchers wish to retrieve investment amount summary of all customers who have joined a certain wealth management product from the cloud server. For this, the bank needs to provide the researchers a decryption key for the corresponding functionality. However, if the encryption scheme used by the bank possesses no function privacy, then the researchers would get to know the specific customer's information from the decryption key provided by the bank. Thus, after performing the assigned computation, for financial gain, the researchers may leak the information to someone. This is clearly undesirable from the privacy point of view. Our function encryption scheme can solve the aforementioned information leak problem, and it can be applied to the following application scenarios such as spam filtering in the mail service, patient record privacy protection in the hospital, and partial decryption of information in the encrypted information store.

1.1. Our Result. The design of our function encryption scheme depends on the idea of key encapsulation. Key encapsulation (also known as hybrid encryption) consists of two parts: an external scheme and an internal scheme. To encrypt messages in a hybrid encryption scheme, a new key is generated for the internal scheme and used to encrypt messages. The key itself is then encrypted using an external scheme. The final hybrid ciphertext consists of two parts: the external ciphertext and the internal ciphertext. If someone wants to decrypt, he should firstly decrypt the external ciphertext to get the key and use this key to decrypt the internal ciphertext. While the internal scheme is symmetric encryption and the external scheme is public key encryption, the hybrid scheme still belongs to public key encryption. Why use the idea of key encapsulation? The external scheme is inefficient, so it is only used to encrypt relatively short keys, while relatively long messages are encrypted by the internal scheme.

We choose symmetric encryption and public key encryption as the internal and external schemes of key encapsulation, respectively. In the design of the function encryption scheme, our idea is as follows: at first, we choose a key k of the internal scheme to encrypt the message x, and then use the external scheme to encrypt the key k to get the two parts of a complete ciphertext. This will allow for decryption during the two-step process described above. First, the key k of the internal scheme is obtained by decrypting the external ciphertext in the external scheme, and then $f(x)$ is obtained by using this key in the internal scheme. Specifically, we describe how to use the public key encryption system, symmetric encryption system, noninteractive proof system, indistinguishable obfuscator, and commitment scheme to design our function encryption scheme. Our scheme achieves indistinguishable security [8].

1.2. Related Works. The concept of function encryption originates from inner product encryption [6] and attribute-based encryption [5]. Early function encryption generally refers to attribute-based encryption, predicate encryption, or inner product encryption. These three kinds of encryption are the preliminary stages of function encryption. Sahai and Waters firstly created the concept of function encryption in 2008 [9]. Later, O'Neill proposed the security definition of the general function encryption [7], which laid the foundation for the research of function encryption.

In 2011, Boneh et al. [3] not only gave the general form of function encryption, but also gave the formal definition and various security definitions of function encryption. In 2012, Gorbunov et al. [10] proposed a nonconcise general-purpose function encryption scheme based on multiparty security calculation. In 2013, Waters [11], Garg et al., [12] and Ananth et al. [13] further improved the efficiency of universal function encryption system. Subsequently, the function encryption scheme for regular language [14] and the function encryption scheme-based deterministic finite automata [15] were proposed. In the same year, Garg et al. [16] constructed a general function encryption scheme for the first time using indistinguishable obfuscation, which set up a new direction for the research of universal function encryption. Goldwasser et al. [17] proposed a simple universal function encryption scheme which uses fully homomorphic encryption, attribute-based encryption, and obfuscation circuit technology as the underlying modules. This scheme is a classical function encryption scheme. Later, Goldwasser et al. [18] firstly proposed multi-input function encryption. They not only proved the adaptive indistinguishable security of the scheme but also proved the simulation-based security of the scheme. It has to be noted that multi-input function encryption scheme is a refinement of function encryption, which can be applied in many occasions and can realize the security processing of multiuser information at different times.

With the deepening of research on function encryption and fully homomorphic encryption, a new full key homomorphic encryption scheme (FKHE) [19] was proposed at the Eurocrypt 2014. It combines fully homomorphic encryption with function encryption. It overcomes the shortcomings of single key shortage of fully homomorphic encryption and provides possibilities for the use of function encryption in the outsourcing computing environment. In 2015, Goyal et al. [20] proposed for the first time the definition and construction of function encryption for random functions, which proved that the construction scheme was simulation-based security. Ananth et al. [13] proved that any selective secure function encryption can be transformed into adaptive secure function encryption and designed an adaptive function encryption scheme that combines public key function encryption with private key function encryption. Brakerski and Segev [21] studied the symmetric function encryption system with function hiding. Abdalla et al. [22] firstly proposed an internal product function encryption scheme based on public key, which set up a new direction of practical internal product function encryption scheme. Bishop et al. [23] firstly proposed the inner product

function encryption scheme of the function hidden in symmetric case. Subsequently, Datta et al. [24] improved Bishop's scheme. In 2018, Kim et al. [25] improved the inner product function encryption scheme of the function hidden to improve its efficiency. They also described three application scenarios of inner product function encryption of the function hidden. Kim et al. made it possible to use the inner product function encryption of the function hidden in practical applications.

1.3. Organization. This paper is organized as follows. The first section is the introduction of this paper. The second section is the existing definition of function encryption. In the third section, we recall some definitions of cryptographic primitives used in scheme construction. In the fourth section, we propose the design of our function encryption. In the fifth section, we prove the security of our constructed scheme. Finally, in the sixth section, we summarize the full text.

1.4. Basic Notation. In the following sections, the security parameter is $\lambda \in \mathbb{N}$. If $|\mathrm{negl}(\lambda)| < 1/\mathrm{poly}(\lambda)$ holds for all polynomials $\mathrm{poly}(\lambda)$ and all sufficiently large λ, we call $\mathrm{negl}(\lambda)$ as negligible function. In this paper, "PPT" represents probabilistic polynomial time and "$x \| y$" represents a concatenation of x and y. Let $\mathscr{F} = \mathscr{F}(\lambda)$ represent an ensemble, each of which is a finite function.

2. Function Encryption

In the second section, the concept of function encryption [3] and its indistinguishable security are rementioned. Indistinguishable security is not only the first consideration of function encryption but also of predicate encryption [4, 6].

2.1. Syntax. For a class of functions $\mathscr{F} = \mathscr{F}(\lambda)$ over message space $\mathscr{M} = \mathscr{M}_\lambda$, a function encryption scheme is composed of the following four algorithms:

(i) $\mathrm{Setup}(1^\lambda) \longrightarrow (\mathrm{mpk}, \mathrm{msk})$: This is a random algorithm. This algorithm requires a security parameter λ as the input of the algorithm, requires the master public key mpk, and the master secret key msk as the output of the algorithm.

(ii) $\mathrm{Enc}(\mathrm{mpk}, m) \longrightarrow \mathrm{CT}$: This algorithm requires the master public key mpk and the plaintext $m \in \mathscr{M}$ as the input of the algorithm and requires a ciphertext CT as the output of the algorithm.

(iii) $\mathrm{KeyGen}(\mathrm{msk}, f) \longrightarrow \mathrm{sk}_f$: This algorithm requires the master secret key msk and the randomized function $f \in \mathscr{F}$ as the input of the algorithm and requires a secret key sk_f as the output of the algorithm.

(iv) $\mathrm{Dec}(\mathrm{sk}_f, \mathrm{CT}) \longrightarrow f(m)$: This algorithm requires the ciphertext CT and the secret key sk_f as the input of the algorithm and requires a string $f(m)$ as the output of the algorithm.

Definition 1 (correctness). If the following conclusion holds for every function $f \in \mathscr{F}$ and every message $m \in \mathscr{M}$, the function encryption scheme for $\mathscr{F}$ is correct.

$$\Pr\Big[(\mathrm{mpk}, \mathrm{msk}) \longleftarrow \mathrm{Setup}\left(1^\lambda\right); \mathrm{Dec}(\mathrm{KeyGen}(\mathrm{msk}, f), \mathrm{Enc}(\mathrm{mpk}, m)) \neq f(m)\Big] = \mathrm{negl}(\lambda). \tag{1}$$

Definition 2 (indistinguishable security of function encryption). The indistinguishable security can be seen as a game between the attacker $\mathscr{A}$ and challenger. This game is divided into five phases.

(i) Setup phase: the challenger generates master key by the Setup algorithm $(\mathrm{mpk}, \mathrm{msk}) \longleftarrow \mathrm{Setup}(1^\lambda)$ and then makes the attacker $\mathscr{A}$ get the master public key mpk.

(ii) Query phase 1: the attacker $\mathscr{A}$ chooses f_i in $\mathscr{F}$ adaptively and makes the challenger get this f_i. The challenger generates the key sk_{f_i} of function f_i by the KeyGen algorithm $\mathrm{sk}_{f_i} \longleftarrow \mathrm{KeyGen}(\mathrm{msk}, f_i)$ and sends it to the attacker. The attacker can repeat this step in any polynomial number of times.

(iii) Challenge phase: the attacker $\mathscr{A}$ chooses two messages $m_0, m_1 \in \mathscr{M}$ such that $f_i(m_0) = f_i(m_1)$ and makes the challenger get it. The Challenger chooses m_b $(b \longleftarrow \{0, 1\})$ from m_0 and m_1, runs the Enc algorithm $\mathrm{CT} \longleftarrow \mathrm{Enc}(\mathrm{mpk}, m_b)$ $(b \longleftarrow \{0, 1\})$, and makes the attacker get the ciphertext CT.

(iv) Query phase 2: key queries continue to be initiated by the attacker $\mathscr{A}$ as before, but it also needs to satisfy that for any query f_i, there is $f_i(m_0) = f_i(m_1)$ holds.

(v) Guess phase: the attacker $\mathscr{A}$ guesses whether the ciphertext CT is encryption for message m_0 or message m_1. Finally, the attacker $\mathscr{A}$ give his guess b' $(b' \longleftarrow \{0, 1\})$.

In this game, the advantage of the attacker $\mathscr{A}$ is $\mathrm{Adv}_{\mathscr{A}} = \Pr[b' = b] - (1/2)$.

3. Preliminaries

In the third section, we recall some concepts of primitives in cryptography, which are used in the construction of function encryption. Here, we omit not only the formal definitions of standard semantically secure public key encryption scheme PKE = (PKE.KeyGen, PKE.Enc, PKE.Dec) but also the formal definitions of standard semantically secure symmetric encryption scheme SE = (SE.Enc, SE.Dec). Next, we describe the formal definitions

of indistinguishable obfuscation, commitment scheme, and witness indistinguishable proof system in detail.

3.1. Indistinguishable Obfuscation. According to the syntax of [16], the concept of indistinguishable obfuscation was recalled.

Definition 3 (Indistinguishable Obfuscation($i\mathcal{O}$)). If the following conclusions hold, then the uniform PPT machine $i\mathcal{O}$ is known as an indistinguishable obfuscator of a circuit class $\{\mathcal{C}_\lambda\}$.

(i) Correctness: for every security parameter $\lambda \in \mathbb{N}$, $C \in \mathcal{C}_\lambda$, and every input x, the following formula holds:

$$\Pr\left[C'(x) = C(x) : C' \longleftarrow i\mathcal{O}(\lambda, C)\right] = 1. \tag{2}$$

(ii) Indistinguishability: there is a negligible function negl that makes the following conclusion hold for every PPT distinguisher Samp, $\mathcal{D}$ (which is not necessarily uniform). For every security parameter $\lambda \in \mathbb{N}$ and every pair of circuits $C_0, C_1 \in \mathcal{C}_\lambda$, if $C_0(x) = C_1(x)$ for every input x, then

$$\left|\Pr\left[D\left(i\mathcal{O}(\lambda, C_0)\right) = 1\right] - \Pr\left[D\left(i\mathcal{O}(\lambda, C_1)\right) = 1\right]\right| \leq \mathrm{negl}(\lambda). \tag{3}$$

3.2. Commitment Scheme. According to the syntax of [26], a commitment scheme Com takes a random number r and a string x as the input and takes $c \longleftarrow \mathrm{Com}(x; r)$ as the output. The following two properties are necessary for a perfectly binding commitment scheme:

(i) Perfectly binding property: this property means that the commitments for two different strings must also be different. More formally, $\forall\ x_1 \neq x_2$ and r_1, r_2, $\mathrm{Com}(x_1; r_1) \neq \mathrm{Com}(x_2; r_2)$.

(ii) Computational hiding property: for every string x_0 and x_1 (the length of x_0 and x_1 is the same) and for every PPT adversary $\mathcal{A}$, the following formula holds:

$$\left|\Pr\left[\mathcal{A}\left(\mathrm{Com}\left(x_0\right)\right) = 1\right] - \Pr\left[\mathcal{A}\left(\mathrm{Com}\left(x_1\right)\right) = 1\right]\right| \leq \mathrm{negl}(\lambda). \tag{4}$$

3.3. Noninteractive Witness Indistinguishable Proof. The syntax of the noninteractive proof system was firstly recalled, and then the formal concept of noninteractive witness indistinguishable (NIWI) proof [27] was also recalled.

3.3.1. Syntax. An efficiently computable relation R is composed of pairs (x, w), in which x and w are named as the statement and the witness, respectively. The language consisting of statements in R is denoted by L. The following three algorithms constitute a noninteractive proof system for a language L:

(i) $\mathrm{NIWI.Setup}(1^\lambda) \longrightarrow \mathrm{crs}$: this algorithm requires a security parameter 1^λ as the input of the algorithm and requires a common reference string crs as the output of the algorithm.

(ii) $\mathrm{NIWI.Prove}(\mathrm{crs}, x, w) \longrightarrow \pi$: this prove algorithm requires the common reference string crs and a statement x along with a witness w as the input of the algorithm. This prove algorithm outputs a proof string π when $(x, w) \in R$ or outputs fail when $(x, w) \notin R$.

(iii) $\mathrm{NIWI.Verify}(\mathrm{crs}, x, \pi) \longrightarrow \{0, 1\}$: this verify algorithm requires the common reference string crs and a statement x with a corresponding proof π as the input. If the proof is valid, this algorithm outputs 1 and otherwise 0.

Definition 4 (NIWI). A noninteractive witness indistinguishable proof system for a language L with a PPT relation R needs to satisfy the following properties:

(i) Perfect completeness property: for all $(x, w) \in R$, the following formula holds:

$$\Pr\left[\mathrm{NIWI.Verify}(\mathrm{crs}, x, \mathrm{NIWI.Prove}(\mathrm{crs}, x, w)) = 1\right] = 1. \tag{5}$$

Here, the reference string crs is generated by the algorithm $\mathrm{crs} \longleftarrow \mathrm{NIWI.Setup}(1^\lambda)$, and the probability is taken over the coins of NIWI.

(ii) Statistical soundness property: for all adversary $\mathcal{A}$, the following formula holds:

$$\Pr\left[\mathrm{NIWI.Verify}(\mathrm{crs}, x, \pi) = 1 \wedge x \notin L \;\middle|\; \begin{array}{l} \mathrm{crs} \longleftarrow \mathrm{NIWI.Setup}(1^\lambda); \\ (x, \pi) \longleftarrow \mathcal{A}(\mathrm{crs}) \end{array}\right] = \mathrm{negl}\left(1^\lambda\right). \tag{6}$$

(iii) Witness indistinguishability property: for any triplet (x, w_0, w_1) ($(x, w_0) \in R$ and $(x, w_1) \in R$), the distributions $\{\mathrm{crs}, \mathrm{NIWI.Prove}(\mathrm{crs}, x, w_0)\}$ and $\{\mathrm{crs}, \mathrm{NIWI.Prove}(\mathrm{crs}, x, w_1)\}$ are computationally indistinguishable (here the reference string crs is generated by the algorithm $\mathrm{crs} \longleftarrow \mathrm{NIWI.Setup}(1^\lambda)$).

4. Construction

In the fourth section, we propose the design of function encryption scheme. In our construction, we set the key space of the symmetric encryption scheme SE to $\{0, 1\}^{\ell_{\mathrm{SE}}}$.

In our scheme, we set the ciphertext length of the public key encryption scheme PKE = (PKE.KeyGen, PKE.Enc, PKE.Dec) to $\mathrm{len}_c = \mathrm{len}_c(1^\lambda)$. In the design of our scheme, the parameter $\mathrm{len} = 2 \cdot \mathrm{len}_c$ will be used, and the symmetric encryption scheme SE needs to satisfy this property:

$$\Pr\left[\mathrm{SE.Dec}(k_1, C') \neq \perp : (k_0, k_1) \longleftarrow \{0,1\}^{\ell_{\mathrm{SE}}},\ k_0 \neq k_1,\ C' \longleftarrow \mathrm{SE.Enc}(k_0, m)\right] \leq \mathrm{negl}(\lambda), \tag{7}$$

where symbol "$\perp$" denotes an error.

In the design of our scheme, the NIWI proof system used is denoted by (NIWI.Setup, NIWI.Prove, NIWI.Verify), the perfectly binding commitment scheme used is denoted by Com, and the indistinguishable obfuscator used is denoted by $i\mathcal{O}$. Now, we begin to give our scheme FE = (Setup, Enc, KeyGen, Dec).

(i) $\mathrm{Setup}(1^\lambda) \longrightarrow (\mathrm{mpk}, \mathrm{msk})$

(1) At first, two pairs of key pairs are generated using the key generation algorithm $(\mathrm{pk}_1, \mathrm{sk}_1) \longleftarrow \mathrm{PKE.KeyGen}(1^\lambda)$ and $(\mathrm{pk}_2, \mathrm{sk}_2) \longleftarrow \mathrm{PKE.KeyGen}(1^\lambda)$ of the public key encryption scheme (note: $(\mathrm{pk}, \mathrm{sk}) \longleftarrow \mathrm{PKE.KeyGen}(1^\lambda)$ is a random algorithm)

(2) Compute a common reference string $\mathrm{crs} \longleftarrow \mathrm{NIWI.Setup}(1^\lambda)$ using the NIWI proof system

(3) Choose a symmetric encryption scheme SE

(4) Compute $C \longleftarrow \mathrm{Com}(0^{\mathrm{len}})$

(5) Let the master public key and the master secret key be $\mathrm{mpk} = (\mathrm{pk}_1, \mathrm{pk}_2, \mathrm{crs}, C, \mathrm{SE})$ and $\mathrm{msk} = \mathrm{sk}_1$, respectively

(ii) $\mathrm{Enc}(\mathrm{mpk}, m) \longrightarrow \mathrm{CT}$:

(1) Choose a secret key K_{SE} of the symmetric encryption and compute the ciphertexts
$\mathrm{CT}_1 \longleftarrow \mathrm{PKE.Enc}(\mathrm{pk}_1, K_{\mathrm{SE}})$
$\mathrm{CT}_2 \longleftarrow \mathrm{PKE.Enc}(\mathrm{pk}_2, K_{\mathrm{SE}})$
$\mathrm{CT}_3 \longleftarrow \mathrm{SE.Enc}(K_{\mathrm{SE}}, m)$

(2) Compute $\pi \longleftarrow \mathrm{NIWI.Prove}(\mathrm{crs}, y, w)$ $(y = (\mathrm{CT}_1, \mathrm{CT}_2, C, \mathrm{pk}_1, \mathrm{pk}_2))$

(i) Either CT_1 and CT_2 are encryptions for the same plaintext, or

(ii) C is a commitment for $\mathrm{CT}_1 \| \mathrm{CT}_2$

The real witness $w_{\mathrm{real}} = (K_{\mathrm{SE}}, r_1, r_2)$ is used to prove the first part of the statement, where the random numbers r_1 and r_2 are used to compute the ciphertexts CT_1 and CT_2, respectively; the trapdoor witness $w_{\mathrm{trap}} = s$ is used to prove the second part of the statement, where the random number s is used to compute C

(3) The ciphertext is $\mathrm{CT} = (\mathrm{CT}_1, \mathrm{CT}_2, \mathrm{CT}_3, \pi)$

(iii) $\mathrm{KeyGen}(\mathrm{msk}, f) \longrightarrow \mathrm{sk}_f$

(1) Compute the decryption key $\mathrm{sk}_f \longleftarrow i\mathcal{O}(\mathcal{G})$ for function f, where the circuit $\mathcal{G}$ is showed in Figure 1; it should be noted that the circuit $\mathcal{G}$ contains the random function f, the secret key sk_1, and the master public key mpk

(iv) $\mathrm{Dec}(\mathrm{sk}_f, \mathrm{CT}) \longrightarrow f(m)$

(1) It inputs CT and decryption key sk_f and outputs $f(m)$

The correctness of the scheme we design is easily derived from the correctness of its components. Next, we will demonstrate the security of this scheme.

5. Proof of Security

Theorem 1. *Assume that the aforementioned function encryption instantiated with a standard semantically secure public key encryption, a standard semantically secure symmetric encryption, a computational hiding commitment, a noninteractive witness indistinguishable proof, and indistinguishably secure obfuscator, it is indistinguishably secure.*

Now, we prove that if the aforementioned assumption is true, no polytime attacker can break our scheme. We will prove indistinguishable security of the function encryption using indistinguishable game. We assume that a polytime attacker $\mathcal{A}$ makes q private key queries. Let f_i $(i \in [q])$ denote the ith function query. There is a constraint $f_i(m_0) = f_i(m_1)$ for each function query.

We need to define a sequence of games to prove theorem 1. The first game is the experiment with the challenge message m_0. Next, we prove that any PPT adversary has almost the same advantage in each game as that of the previous game.

Game 1. The challenger encrypts message m_0 in the challenge ciphertext.

(i) *Setup Phase.* The challenger firstly computes the key pair $(\mathrm{mpk}, \mathrm{msk}) \longleftarrow \mathrm{Setup}(1^\lambda)$ and gives master public mpk to $\mathcal{A}$. Choose a symmetric encryption scheme SE and three secret keys $k_1 \longleftarrow \{0,1\}^{\ell_{\mathrm{SE}}}$, $k_0 \longleftarrow \{0,1\}^{\ell_{\mathrm{SE}}}$, and $k_2 \longleftarrow \{0,1\}^{\ell_{\mathrm{SE}}}$ for the symmetric encryption scheme. Compute the ciphertexts:

$$\begin{aligned}
&\mathrm{CT}^*_{1,1} \longleftarrow \mathrm{PKE.Enc}(\mathrm{pk}_1, k_1),\\
&\mathrm{CT}^*_{2,1} \longleftarrow \mathrm{PKE.Enc}(\mathrm{pk}_2, k_1).\\
&\mathrm{CT}^*_{1,0} \longleftarrow \mathrm{PKE.Enc}(\mathrm{pk}_1, k_0),\\
&\mathrm{CT}^*_{2,0} \longleftarrow \mathrm{PKE.Enc}(\mathrm{pk}_2, k_0).\\
&\mathrm{CT}^*_{1,2} \longleftarrow \mathrm{PKE.Enc}(\mathrm{pk}_1, k_2),\\
&\mathrm{CT}^*_{2,2} \longleftarrow \mathrm{PKE.Enc}(\mathrm{pk}_2, k_2).
\end{aligned}$$

Set $c_1 = \mathrm{CT}^*_{1,1} \| \mathrm{CT}^*_{2,1}$, $c_2 = \mathrm{CT}^*_{1,0} \| \mathrm{CT}^*_{2,0}$, $c_3 = \mathrm{CT}^*_{1,2} \| \mathrm{CT}^*_{2,2}$, $c_4 = \mathrm{CT}^*_{1,1} \| \mathrm{CT}^*_{2,0}$, $c_5 = \mathrm{CT}^*_{1,2} \| \mathrm{CT}^*_{2,0}$.

(ii) *Query Phase.* The adversary $\mathcal{A}$ submits query of f adaptively. The challenger sends $\mathrm{sk}_f \longleftarrow \mathrm{KeyGen}(\mathrm{msk}, f)$ to $\mathcal{A}$.

(iii) *Challenge Phase.* The challenger chooses a challenge plaintext m_0 from two plaintexts and a secret key k_1 of the symmetric encryption, computes the ciphertexts $\mathrm{CT}^*_1 \longleftarrow \mathrm{PKE.Enc}(\mathrm{pk}_1, k_1)$, $\mathrm{CT}^*_2 \longleftarrow \mathrm{PKE.Enc}(\mathrm{pk}_2, k_1)$, and $\mathrm{CT}^*_3 \longleftarrow \mathrm{SE.Enc}(k_1, m_0)$. Compute a NIWI proof $\pi^* \longleftarrow \mathrm{NIWI.Prove}(\mathrm{crs}, y^*, w)$ $(y^* = (\mathrm{CT}^*_1, \mathrm{CT}^*_2, C, \mathrm{pk}_1, \mathrm{pk}_2))$. Set $\mathrm{st} = (\mathrm{CT}^*_1, \mathrm{CT}^*_2, k_1, \pi^*)$. At last, it returns $\mathrm{CT}^* = (\mathrm{CT}^*_1, \mathrm{CT}^*_2, \mathrm{CT}^*_3, \pi^*)$.

Input: Ciphertext CT
Constants: Master public key mpk, secret key sk_1, random function f

(1) Parse $CT = (CT_1, CT_2, CT_3, \pi)$.
(2) If NIWI.Verify $(crs, y, \pi) = 0$, output $\perp$ and stop. If NIWI.Verify $(crs, y, \pi) = 1$, continue to the next step. Here the statement corresponding to π is $y = (CT_1, CT_2, C, pk_1, pk_2)$.
(3) Compute $K_{SE} \longleftarrow$ PKE.Dec (sk_1, CT_1), $m \longleftarrow$ SE.Dec (K_{SE}, CT_3).
(4) Output $f(m)$.

Figure 1: Functionality $\mathcal{G}$.

Game 2. This game is exactly the same as the previous game (Game 1) except for a little difference. The difference is that for all key queries of f, the corresponding secret key sk_f is computed by $sk_f \longleftarrow i\mathcal{O}(\mathcal{G}_f)$, where the circuit $\mathcal{G}_f$ is described in Figure 2.

Game 3. This game is exactly the same as the previous game (Game 2) except for a little difference. The difference is that the commitment C is computed as follows: the challenge ciphertext is denoted by $CT^* = (CT_1^*, CT_2^*, CT_3^*, \pi^*)$ and $C \longleftarrow \mathrm{Com}(CT_1^* \| CT_2^*)$.

Game 4. This game is exactly the same as the previous game (Game 3) except for a little difference. The difference is that in every challenge ciphertext $CT^* = (CT_1^*, CT_2^*, CT_3^*, \pi^*)$ π^* is computed using the trapdoor witness.

Game 5. This game is exactly the same as the previous game (Game 4) except for a little difference. The difference is that we modify the challenge ciphertext $CT^* = (CT_1^*, CT_2^*, CT_3^*, \pi^*)$: the second ciphertext CT_2^* is an encryption of k_0, that is, $CT_2^* \longleftarrow$ PKE.Enc (pk_2, k_0).

Game 6. This game is exactly the same as the previous game (Game 5) except for a little difference. The difference is that for all key queries of f, the corresponding secret key sk_f is computed by $sk_f \longleftarrow i\mathcal{O}(\mathcal{G}_f^*)$, where the circuit $\mathcal{G}_f^*$ is described in Figure 3.

Game 7. This game is exactly the same as the previous game (Game 6) except for a little difference. The difference is that we modify the challenge ciphertext $CT^* = (CT_1^*, CT_2^*, CT_3^*, \pi^*)$: the first ciphertext CT_1^* is an encryption of k_0, that is, $CT_1^* \longleftarrow \mathrm{PKE.Enc}(pk_1, k_0)$.

Game 8. This game is exactly the same as the previous game (Game 7) except for a little difference. The difference is that the symmetric encryption key k_1 is replaced by another key k_2.

Game 9. This game is exactly the same as the previous game (Game 8) except for a little difference. The difference is that we modify the challenge ciphertext $CT^* = (CT_1^*, CT_2^*, CT_3^*, \pi^*)$: the third ciphertext CT_3^* is an encryption of m_1, that is, $CT_3^* \longleftarrow \mathrm{SE.Enc}(k_2, m_1)$.

Game 10. This game is exactly the same as the previous game (Game 9) except for a little difference. The difference is that we modify the challenge ciphertext $CT^* = (CT_1^*, CT_2^*, CT_3^*, \pi^*)$: the first ciphertext CT_1^* is an encryption of k_2, that is $CT_1^* \longleftarrow \mathrm{PKE.Enc}(pk_1, k_2)$.

Game 11. This game is exactly the same as the previous game (Game 10) except for a little difference. The difference is that for all key queries f, the corresponding secret key sk_f is computed by $sk_f \longleftarrow i\mathcal{O}(\mathcal{G}_f)$.

Game 12. This game is exactly the same as the previous game (Game 11) except for a little difference. The difference is that we modify the challenge ciphertext $CT^* = (CT_1^*, CT_2^*, CT_3^*, \pi^*)$: the second ciphertext CT_2^* is an encryption of k_2, that is, $CT_2^* \longleftarrow \mathrm{PKE.Enc}(pk_2, k_2)$.

Game 13. This game is exactly the same as the previous game (Game 12) except for a little difference. The difference is that in every challenge ciphertext $CT^* = (CT_1^*, CT_2^*, CT_3^*, \pi^*)$, the proof π^* is computed using the real witness.

Game 14. This game is exactly the same as the previous game (Game 13) except for a little difference. The difference is that the commitment C is computed as follows: $C \longleftarrow \mathrm{Com}(0^{\mathrm{len}})$. Note that this game corresponds to the selective indistinguishable game that has been honestly executed, where the challenger encrypts the message m_1 in the challenge ciphertext.

The description for the sequence of games is completed. Next, we prove that the neighbouring games are indistinguishable.

Lemma 1. *Game 1 and Game 2 are computationally indistinguishable when $i\mathcal{O}$ is an indistinguishable obfuscator.*

Proof. We can see that the difference between Game 1 and Game 2 lies only in the calculation method of the secret key sk_f. In the former experiment, Game 1, for any key query f, the secret key sk_f is outputted by $i\mathcal{O}(\mathcal{G})$; however, in the latter experiment, Game 2, the secret key sk_f is outputted by $i\mathcal{O}(\mathcal{G}_f)$. If we want to prove that Game 1 and Game 2 are computationally indistinguishable, we need to prove that for any input CT, $\mathcal{G}$ and $\mathcal{G}_f$ have identical output for identical input. Then, according to the security of indistinguishable obfuscator, we can get that $i\mathcal{O}(\mathcal{G})$ and $i\mathcal{O}(\mathcal{G}_f)$ are computationally indistinguishable, which means that Game 1 and Game 2 are computationally indistinguishable.

First, we will prove that for any input $CT = (CT_1, CT_2, CT_3, \pi)$, $\mathcal{G}$ outputs $\perp$ if and only if $\mathcal{G}_f$ outputs $\perp$.

We can find that both $\mathcal{G}$ and $\mathcal{G}_f$ output $\perp$ if and only if the proof π is invalid, that is, NIWI.Verify $(crs, y, \pi) = 0$. Here, $y = (CT_1, CT_2, C, pk_1, pk_2)$. If the proof π is valid, we define that an input $CT = (CT_1, CT_2, CT_3, \pi)$ is valid.

Next, we prove that for any valid input $CT = (CT_1, CT_2, CT_3, \pi)$, $\mathcal{G}(CT) = \mathcal{G}_f(CT)$.

Input: Ciphertext CT
Constants: sk_1, mpk, f, k_0, k_1, k_2, c_1, c_2, c_3, c_4, c_5

(1) Parse $\mathrm{CT} = (\mathrm{CT}_1, \mathrm{CT}_2, \mathrm{CT}_3, \pi)$.
(2) If NIWI.Verify (crs, y, π) = 0, output $\perp$ and stop. If NIWI.Verify (crs, y, π) = 1, continue to the next step. Here the statement corresponding to π is $y = (\mathrm{CT}_1, \mathrm{CT}_2, C, \mathrm{pk}_1, \mathrm{pk}_2)$.
(3) If there exists a $c_i = \mathrm{CT}_1^* \,||\, \mathrm{CT}_2^*$, then if SE.Dec $(k_0, \mathrm{CT}_3) = m$, and $m \neq \perp$, output $f(m)$ and stop, else if SE.Dec $(k_1, \mathrm{CT}_3) = m$, and $m \neq \perp$, output $f(m)$ and stop, else SE.Dec $(k_2, \mathrm{CT}_3) = m$, and $m \neq \perp$, output $f(m)$ and stop, otherwise continue to the next step.
(4) Compute $K_{\mathrm{SE}} \longleftarrow$ PKE.Dec $(\mathrm{sk}_1, \mathrm{CT}_1)$, $m \longleftarrow$ SE.Dec $(K_{\mathrm{SE}}, \mathrm{CT}_3)$.
(5) Output $f(m)$.

Figure 2: Functionality $\mathcal{G}_f$.

Input: Ciphertext CT
Constants: sk_2, mpk, f, k_0, k_1, k_2, c_1, c_2, c_3, c_4, c_5

(1) Parse $\mathrm{CT} = (\mathrm{CT}_1, \mathrm{CT}_2, \mathrm{CT}_3, \pi)$.
(2) If NIWI.Verify (crs, y, π) = 0, output $\perp$ and stop. If NIWI.Verify (crs, y, π) = 1, continue to the next step. Here the statement corresponding to π is $y = (\mathrm{CT}_1, \mathrm{CT}_2, C, \mathrm{pk}_1, \mathrm{pk}_2)$.
(3) If there exists a $c_i = \mathrm{CT}_1^* \,||\, \mathrm{CT}_2^*$, then if SE.Dec $(k_0, \mathrm{CT}_3) = m$, and $m \neq \perp$, output $f(m)$ and stop, else if SE.Dec $(k_1, \mathrm{CT}_3) = m$, and $m \neq \perp$, output $f(m)$ and stop, else SE.Dec $(k_2, \mathrm{CT}_3) = m$, and $m \neq \perp$, output $f(m)$ and stop, otherwise continue to the next step.
(4) Compute $K_{\mathrm{SE}} \longleftarrow$ PKE.Dec $(\mathrm{sk}_2, \mathrm{CT}_2)$, $m \longleftarrow$ SE.Dec $(K_{\mathrm{SE}}, \mathrm{CT}_3)$.
(5) Output $f(m)$.

Figure 3: Functionality $\mathcal{G}_f^*$.

Here we have to consider two cases:

(i) *Case 1*. There does not exist a c_i, such that $c_i = \mathrm{CT}_1^* \| \mathrm{CT}_2^*$.
(ii) *Case 2*. There exists a c_i, such that $c_i = \mathrm{CT}_1^* \| \mathrm{CT}_2^*$.

For the case 1, both $\mathcal{G}$ and $\mathcal{G}_f$ compute k_1 with sk_1 by decrypting CT_1, compute m with k_1 by decrypting CT_3, and outputs $f(m)$. In the case 2, $\mathcal{G}$ computes k_1 with sk_1 by decrypting CT_1, computes m with k_1 by decrypting CT_3, and also outputs $f(m)$. What's more, because $c_i = \mathrm{CT}_1^* \| \mathrm{CT}_2^*$, $\mathcal{G}_f$ also outputs $f(m)$. So we can get that $\mathcal{G}(\mathrm{CT}) = \mathcal{G}_f(\mathrm{CT})$.

The experiment $G_{2,i}$ $(0 \leq i \leq q)$ is defined as follows: in $G_{2,i}$, the first ith queries are answered by $\mathcal{G}$ and the remaining $(i+1)$th to qth queries are answered by $\mathcal{G}_f$. We can see that $G_{2,0}$ happens to be Game 1 and $G_{2,q}$ happens to be Game 2.

Here, we prove that if there is a PPT adversary $\mathcal{A}$ which can distinguish the experiments $G_{2,i}$ and $G_{2,i+1}$ with nonnegligible advantage, there is another PPT adversary $\mathcal{B}$ which can break the security of indistinguishable obfuscator with nonnegligible advantage.

We construct $\mathcal{B}$ by $\mathcal{A}$ as follows:

(1) First of all, the adversary $\mathcal{B}$ honestly runs the Game 1.
(2) For the first ith key queries f, the adversary $\mathcal{B}$ computes the key sk_f by $\mathcal{G}_f$. For the remaining $(i+2)$th to qth key queries f, the adversary $\mathcal{B}$ computes the key sk_f by $\mathcal{G}$.
(3) For the $(i+1)$th key queries f, the adversary $\mathcal{B}$ respectively constructs the circuit $\mathcal{G}$ and $\mathcal{G}_f$ and sends them to the challenger of the $i\mathcal{O}$ and receives an obfuscation sk_f. Then, the adversary $\mathcal{B}$ gives sk_f to $\mathcal{A}$.
(4) The adversary $\mathcal{B}$ runs the rest of the experiment according to the Game 1.
(5) At last, the adversary $\mathcal{B}$ gives the output of the game to the adversary $\mathcal{A}$ and the obfuscation challenger.

We happen to be in experiment $G_{2,i}$ when the challenger of $i\mathcal{O}$ chose the circuit $\mathcal{G}$; We happen to be in experiment $G_{2,i+1}$ when the challenger of $i\mathcal{O}$ chose the circuit $\mathcal{G}_f$. Therefore, if the adversary $\mathcal{A}$ can distinguish between the two experiments with nonnegligible advantage, then the adversary $\mathcal{B}$ can break the security of indistinguishable obfuscator with the same advantage.

In summary, Game 1 is computationally indistinguishable from Game 2. □

Lemma 2. *If Com is a computationally hiding commitment scheme, Game 2 is computationally indistinguishable from Game 3.*

Proof. We can see that the difference between Game 2 and Game 3 lies only in the calculation method of the commitment C. In the former game, Game 2, C is a commitment for 0^{len}; however, in the latter game, Game 3, C is a commitment for $\mathrm{CT}_1^* \| \mathrm{CT}_2^*$. It is important to note that the random number used to compute C is never used anywhere.

Therefore, the property of computational hiding in the commitment guarantees that Game 2 is computationally indistinguishable from Game 3. □

Lemma 3. *Because of witness indistinguishability of NIWI, Game 3 is computationally indistinguishable from Game 4.*

Proof. We can see that the difference between Game 3 and Game 4 lies only in the witness w used. In Game 3, for proving that CT_1^* and CT_2^* are encryptions of the same message, we use the real witness to compute π. However, in Game 4, for proving that C is a commitment for $\mathrm{CT}_1^* \| \mathrm{CT}_2^*$, we use the trapdoor witness to compute π. Since NIWI is witness indistinguishable, Game 3 is computationally indistinguishable from Game 4. □

Lemma 4. *If* (*PKE.KeyGen*, *PKE.Enc*, *PKE.Dec*) *is a semantically secure public key encryption scheme, Game 4 is computationally indistinguishable from Game 5.*

Proof. We can see that the difference between Game 4 and Game 5 lies only in the calculation method of the second ciphertexts CT_2^* in the challenge ciphertexts CT^*. In Game 4, CT_2^* is an encryption of the random message k_1, while in Game 5, CT_2^* is an encryption of the random message k_0. Next, we show that if there is an adversary $\mathscr{A}$ who can distinguish Game 4 from Game 5, there is an adversary $\mathscr{B}$ who can break the semantic security of the public key encryption system. The adversary $\mathscr{B}$ is designed as follows:

(1) At first, the adversary $\mathscr{B}$ received a public key pk from the challenger.

(2) $\mathscr{B}$ generates $(\mathrm{pk}_1, \mathrm{sk}_1) \longleftarrow \mathrm{PKE.KeyGen}(1^\lambda)$, $\mathrm{crs} \longleftarrow \mathrm{NIWI.Setup}(1^\lambda)$, and chooses a symmetric encryption scheme SE. Next, the adversary $\mathscr{B}$ encrypts the string k_1 using pk_1 to compute the ciphertext CT_1^*.

(3) The adversary $\mathscr{B}$ sends (k_1, k_0) to the challenger and receives a ciphertext CT_2^*. Then, $\mathscr{B}$ computes the commitment $C = \mathrm{Com}(\mathrm{CT}_1^* \| \mathrm{CT}_2^*)$.

(4) The adversary $\mathscr{B}$ runs the rest of the experiment according to Game 4 and Game 5.

(5) At last, the adversary $\mathscr{A}$ received the output of the experiment from the adversary $\mathscr{B}$.

(6) If adversary $\mathscr{A}$ outputs Game 4, the adversary $\mathscr{B}$ outputs that CT_2^* is an encryption of k_1, otherwise outputs that CT_2^* is an encryption of k_0.

We can see that the adversary $\mathscr{B}$ just runs Game 4 when CT_2^* is an encryption of k_1, and $\mathscr{B}$ just runs Game 5 when CT_2^* is an encryption of k_0. Therefore, if there exists an adversary who can distinguish the outputs of the two games with nonnegligible advantage, we can construct another adversary who can break the semantic security of public key encryption with nonnegligible advantage. □

Lemma 5. *If Com is perfectly binding, and NIWI is statistically sound, $i\mathcal{O}$ is an indistinguishable obfuscator and Game 5 and Game 6 are computationally indistinguishable.*

Proof. Similar to the proof of Lemma 1, we firstly prove that for any input $\mathrm{CT} = (\mathrm{CT}_1, \mathrm{CT}_2, \mathrm{CT}_3, \pi)$, $\mathscr{G}_f$ outputs $\perp$ if and only if $\mathscr{G}_f^*$ outputs $\perp$.

We can see that both $\mathscr{G}_f$ and $\mathscr{G}_f^*$ output $\perp$ if and only if the proof π is invalid, that is, $\mathrm{NIWI.Verify}(\mathrm{crs}, y, \pi) = 0$ ($y = (\mathrm{CT}_1, \mathrm{CT}_2, C, \mathrm{pk}_1, \mathrm{pk}_2)$).

So we need to consider valid inputs only. Then, we will prove that all valid inputs $\mathrm{CT} = (\mathrm{CT}_1, \mathrm{CT}_2, \mathrm{CT}_3, \pi)$ meet one of the following properties:

(i) CT_1 and CT_2 are encryptions for the same message

(ii) There exists an i such that $\mathrm{CT}_1 \| \mathrm{CT}_2 = c_i$

We are going to use a proof by contradiction here. We assume that there is a valid input that satisfies neither of the above properties. Because NIWI is statistically sound, the statement $y = (\mathrm{CT}_1, \mathrm{CT}_2, C, \mathrm{pk}_1, \mathrm{pk}_2)$ must have either a real witness or a trapdoor witness when the input is valid. However, because CT_1 and CT_2 are encryptions of different messages, the real witness does not exist. Therefore, there must be a trapdoor witness to make the input valid. That means there is s such that $C = \mathrm{Com}(\mathrm{CT}_1 \| \mathrm{CT}_2; s)$. On the other hand, since $C = \mathrm{Com}(\mathrm{CT}_1^* \| \mathrm{CT}_2^*; s)$ and Com is perfectly binding, $\mathrm{CT}_1 \| \mathrm{CT}_2 = \mathrm{CT}_1^* \| \mathrm{CT}_2^* = c_5$. Thus, we get a contradiction, and the assumption is not true.

Next, we will prove that for any input $\mathrm{CT} = (\mathrm{CT}_1, \mathrm{CT}_2, \mathrm{CT}_3, \pi)$, $\mathscr{G}_f(\mathrm{CT}) = \mathscr{G}_f^*(\mathrm{CT})$.

If both CT_1 and CT_2 are encryptions for the same message, then $\mathrm{PKE.Dec}(\mathrm{sk}_1, \mathrm{CT}_1) = \mathrm{PKE.Dec}(\mathrm{sk}_2, \mathrm{CT}_2) = k_1$, and $\mathrm{SE.Dec}(k_1, \mathrm{CT}_3) = m$. Therefore, both G_f and G_f^* output $f(m)$. On the other hand, if there is an i satisfying $\mathrm{CT}_1 \| \mathrm{CT}_2 = c_i$, then both G_f and G_f^* output $f(m)$. Therefore, for all valid inputs, G_f and G_f^* have same output for the same input, that is, $\mathscr{G}_f(\mathrm{CT}) = \mathscr{G}_f^*(\mathrm{CT})$.

In summary, if there exists an adversary which can distinguish the outputs of these games with nonnegligible advantage, we can construct another adversary which can break the security of indistinguishable obfuscation with the same advantage. □

Lemma 6. *Game 6 is computationally indistinguishable from Game 7, when PKE is a semantically secure public key encryption scheme.*

Proof. The proof method of this lemma is the same as the proof method of lemma 4, so it is omitted here. □

Lemma 7. *Game 7 is computationally indistinguishable from Game 8.*

Proof. We can see that Game 7 and Game 8 differ in the secret key used in the symmetric encryption scheme. In Game 7, k_1 is used to encrypt m_0, while in Game 8, k_2 is used to encrypt m_0. We can find that both k_1 and k_2 are chosen

randomly from the key space of symmetric encryption scheme. So Game 7 is computationally indistinguishable from Game 8. □

Lemma 8. *The outputs of Game 8 and Game 9 are computationally indistinguishable when* (*SE.KeyGen*, *SE.Enc*, *SE.Dec*) *is a semantically secure symmetric encryption scheme.*

Proof. Since the proof method of this lemma is the same as that of lemma 4, the proof process is omitted here. □

Lemma 9. *The outputs of Game 9 and Game 10 are computationally indistinguishable when* (*PKE.KeyGen*, *PKE.Enc*, *PKE.Dec*) *is a semantically secure public key encryption scheme.*

Proof. Since the proof method of this lemma is the same as that of lemma 4, the proof process is omitted here. □

Lemma 10. *Game 10 and Game 11 are computationally indistinguishable when NIWI is statistically sound, Com is perfectly binding, and $i\mathcal{O}$ is an indistinguishable obfuscator.*

Proof. Since the proof method of this lemma is the same as that of lemma 5, the proof process is omitted here. □

Lemma 11. *The outputs of Game 11 and Game 12 are computationally indistinguishable when* (*PKE.KeyGen*, *PKE.Enc*, *PKE.Dec*) *is a semantically secure public key encryption scheme.*

Proof.
Since the proof method of this lemma is the same as that of lemma 4, the proof process is omitted here. □

Lemma 12. *Because of witness indistinguishability of NIWI, Game 12 and Game 13 are computationally indistinguishable.*

Proof. Since the proof method of this lemma is the same as that of lemma 3, the proof process is omitted here. □

Lemma 13. *Game 13 is computationally indistinguishable from Game 14 when Com is a computationally hiding commitment scheme.*

Proof. Since the proof method of this lemma is the same as that of lemma 2, the proof process is omitted here. □

6. Conclusion

We firstly design a function encryption scheme using the key encapsulation mechanism in this paper. This mechanism combines public key encryption with symmetric encryption. We encrypt the message using symmetric encryption, and then we encrypt the key of symmetric encryption using public key encryption. In the construction of function encryption scheme, we use noninteractive witness indistinguishable proof, commitment scheme, and indistinguishable obfuscator. Finally, the indistinguishable security of the designed function encryption is proven.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 11401172) and Science and Technology Project of Henan Educational Committee of China (No. 20A520012).

References

[1] G. Craig, *A Fully Homomorphic Encryption Scheme*, Stanford University, Stanford, CA, USA, 2009.

[2] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proceedings of the 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 457–473, Aarhus, Denmark, May 2005.

[3] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: definitions and challenges," in *Proceedings of the 8th Theory of Cryptography Conference, TCC 2011*, pp. 253–273, Providence, RI, USA, March 2011.

[4] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Proceedings of the 4th Theory of Cryptography Conference, TCC 2007*, pp. 535–554, Amsterdam, The Netherlands, February 2007.

[5] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006*, pp. 89–98, Alexandria, VA, USA, October-November 2006.

[6] J. Katz, A. Sahai, and B. Waters, "Predicate encryption supporting disjunctions, polynomial equations, and inner products," in *Proceedings of the 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 146–162, Istanbul, Turkey, April 2008.

[7] A. O'Neill, "Definitional issues in functional encryption," Cryptology ePrint Archive: Report 2010/556, 2010.

[8] S. Goldwasser and M. Bellare, *Lecture Notes on Cryptography*, MIT Laboratory of Computer Science, Cambridge, MA, USA, 2001.

[9] A. Sahai and B. Waters, Slides on Functional Encryption, PowerPoint Presentation, 2008.

[10] S. Gorbunov, V. Vaikuntanathan, and H. Wee, "Functional encryption with bounded collusions via multi-party computation," in *Proceedings of the 32nd Annual Cryptology Conference*, pp. 162–179, Santa Barbara, CA, USA, August 2012.

[11] B. Waters, "A punctured programming approach to adaptively secure functional encryption," in *Proceedings of the 35th Annual Cryptology Conference*, pp. 678–697, Santa Barbara, CA, USA, August 2015.

[12] S. Garg, C. Gentry, S. Halevi, and M. Zhandry, "Functional encryption without obfuscation," in *Proceedings of the 13th International Conference on Theory of Cryptography, TCC 2016*, pp. 480–511, Tel Aviv, Israel, January 2016.

[13] P. Ananth, Z. Brakerski, G. Segev, and V. Vaikuntanathan, "From selective to adaptive security in functional encryption," in *Proceedings of the 35th Annual Cryptology Conference*, pp. 657–677, Santa Barbara, CA, USA, August 2015.

[14] B. Waters, "Functional encryption for regular languages," in *Proceedings of the 32nd Annual Cryptology Conference*, pp. 218–235, Santa Barbara, CA, USA, August 2012.

[15] S. C. Ramanna, "DFA-based functional encryption: adaptive security from dual system encryption," IACR Cryptology ePrint Archive: Report 2013/638, 2013.

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

[17] S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich, "Reusable garbled circuits and succinct functional encryption," in *Proceedings of the Symposium on Theory of Computing Conference, STOC'13*, pp. 555–564, Palo Alto, CA, USA, June 2013.

[18] S. Goldwasser, V. Goyal, A. Jain, and A. Sahai, "Multi-input functional encryption," IACR Cryptology ePrint Archive Report: 2013/727, 2013.

[19] D. Boneh, C. Gentry, S. Gorbunov et al., "Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits," in *Proceedings of the 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 533–556, Copenhagen, Denmark, May 2014.

[20] V. Goyal, A. Jain, V. Koppula, and A. Sahai, "Functional encryption for randomized functionalities," in *Proceedings of the 12th Theory of Cryptography Conference, TCC 2015*, pp. 325–351, Warsaw, Poland, March 2015.

[21] Z. Brakerski and G. Segev, "Function-private functional encryption in the private-key setting," in *Proceedings of the 12th Theory of Cryptography Conference, TCC 2015*, pp. 306–324, Warsaw, Poland, March 2015.

[22] M. Abdalla, F. Bourse, A. D. Caro, and D. Pointcheval, "Simple functional encryption schemes for inner products," in *Proceedings of the 18th IACR International Conference on Practice and Theory in Public-Key Cryptography*, pp. 733–751, Gaithersburg, MD, USA, March-April 2015.

[23] A. Bishop, A. Jain, and L. Kowalczyk, "Function-hiding inner product encryption," in *Proceedings of the 21st International Conference on the Theory and Application of Cryptology and Information Security*, pp. 470–491, Auckland, New Zealand, November-December 2015.

[24] P. Datta, R. Dutta, and S. Mukhopadhyay, "Functional encryption for inner product with full function privacy," in *Proceedings of the 19th IACR International Conference on Practice and Theory in Public-Key Cryptography*, pp. 164–195, Taipei, Taiwan, March 2016.

[25] S. Kim, K. Lewi, A. Mandal, H. Montgomery, A. Roy, and D. J. Wu, "Function-hiding inner product encryption is practical," in *Proceedings of the 11th International Conference on Security and Cryptography for Networks, SCN 2018*, pp. 544–562, Amalfi, Italy, September 2018.

[26] O. Goldreich, *Foundations of Cryptographyčž Basic Tools*, The Press Syndicate of the University of Cambridge, Cambridge, UK, 2001.

[27] U. Feige and A. Shamir, "Witness indistinguishable and witness hiding protocols," in *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pp. 416–426, Baltimore, MA, USA, May 1990.