International Journal of Networking and Computing – www.ijnc.org, ISSN 2185-2847
Volume 11, Number 2, pages 299–318, July 2021

Public-Key Projective Arithmetic Functional Encryption

Shingo Hasegawa
Graduate School of Information Sciences
Tohoku University
41 Kawauchi, Aoba-ku, Sendai, Miyagi, 980–8576, Japan

Masashi Hisai
Cyber Physical System Security Department
Panasonic Corporation
Kadoma, Osaka, Japan

and

Hiroki Shizuya
Graduate School of Information Sciences
Tohoku University
41 Kawauchi, Aoba-ku, Sendai, Miyagi, 980–8576, Japan

Received: February 12, 2021
Revised: April 28, 2021
Accepted: June 14, 2021
Communicated by Yasuyuki Nogami

Abstract

Ananth and Sahai proposed the *projective arithmetic functional encryption* (PAFE) and showed that PAFE derives a single-key selective secure functional encryption with the help of the randomizing polynomials scheme (RP), namely PAFE with RP achieves the indistinguishability obfuscation (iO). Their PAFE considers a secret-key type functional encryption only and a public-key counterpart is not known.

We propose the public-key version: pkPAFE, and show that pkPAFE with RP derives a public-key functional encryption which is single-key selective secure. This means that our pkPAFE achieves iO as well as the original PAFE by Ananth and Sahai.

Keywords: Projective Arithmetic Functional Encryption, Randomizing Polynomials Scheme, Indistinguishability Obfuscation

1 Introduction

An obfuscator is the complier which takes a program and outputs another program such that the resulting program maintains the functionality and its internal execution or code is obfuscated. The formal definition of the obfuscator was firstly given by Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan, Yang [5]. They proposed the notion of the natural and ideal obfuscator, the virtual black-box obfuscation (VBB), however, they showed that the VBB cannot be implemented for any circuits.

The indistinguishability obfuscator ($\mathsf{i}\mathcal{O}$) [5, 10] is a weaker notion of obfuscator rather than the VBB. The $\mathsf{i}\mathcal{O}$ only requires that for the obfuscations of two circuits of the same size and functionality, the distributions of outputs of them are computationally indistinguishable. Although the $\mathsf{i}\mathcal{O}$ does not satisfy the ideal definition of obfuscator, a candidate of $\mathsf{i}\mathcal{O}$ was firstly found by Garg, Gentry, Halevi, Raykova, Sahai and Waters [9]. Moreover, it is shown that $\mathsf{i}\mathcal{O}$ can derive many other cryptographic primitives such as the public-key encryption and the digital signature [14] [7]. Therefore, $\mathsf{i}\mathcal{O}$ becomes one of the most attractive cryptographic primitives in the modern cryptography, and constructing $\mathsf{i}\mathcal{O}$ from the standard assumptions or primitives is an important open problem.

One of the main strategies to construct $\mathsf{i}\mathcal{O}$ is to employ the functional encryption [8]. In the functional encryption, the decryption algorithm outputs the circuit value $C(x)$ of the message x for the circuit C from the ciphertext CT, whereas the decryption algorithm outputs the message x in the ordinary encryption. There are several results for constructing $\mathsf{i}\mathcal{O}$ from the functional encryption [2, 6, 11, 12, 13]. In [6], Bitansky and Vaikuntanathan showed that $\mathsf{i}\mathcal{O}$ can be derived from a public-key single-key selective secure functional encryption for $\mathbf{NC^1}$ which is the class of polynomial-size and logarithmic-depth circuits. The public-key single-key selective secure functional encryption can be constructed from the secret-key single-key selective secure functional encryption [11]. These results suggest that a single-key selective secure functional encryption serves as an interface which connects $\mathsf{i}\mathcal{O}$ to other cryptographic primitives. In fact, Lin and Tessaro [13] showed that a single-key selective secure functional encryption for $\mathbf{NC^1}$ can be converted from a fully selective secure functional encryption for $\mathbf{NC^0}$ which is the class of polynomial-size and constant-depth circuits, for public-key and secret-key cases. Their result suggests that the strength of the security of functional encryption can relax the target class of circuits of functional encryption.

Ananth and Sahai [3] presented another direction of the research. They proposed the projective arithmetic functional encryption (PAFE) and showed that PAFE derives a single-key selective secure functional encryption. PAFE differs from the functional encryption in its decryption procedure. In PAFE, the ciphertext is *partially* decrypted by the projective decryption algorithm. Then multiple partially decrypted values are combined to retrieve the circuit value by the recover algorithm. We note that the conversion from PAFE to a single-key FE can be done for any class of circuits although PAFE is instantiated for $\mathbf{NC^0}$. The result of [3] considered the secret-key type functional encryption only, whereas [6] deals with both the secret-key and the public-key functional encryption.

In this paper we consider PAFE in the public-key setting. In the previous paper [3], the public-key version of PAFE is mentioned, however, the construction is not given. Therefore we aim to show the construction. We give a definition and a syntax of the *public-key* projective arithmetic functional encryption (pkPAFE). We also show that pkPAFE can derive a single-key selective secure public-key functional encryption with the help of the randomizing polynomials scheme (RP) [4][1], as in the secret-key case of [3]. RP is a conversion protocol which outputs a sequence of polynomials from an input circuit such that the output of the target circuit is maintained to polynomials.

We briefly describe here our conversion from pkPAFE to a public-key FE. The resulting public-key FE consists of four algorithms, setup, key generation, encryption and decryption. The setup algorithm coincides with the one of pkPAFE. It outputs a pair of a public key and a secret key. The key generation computes a functional key for the input circuit. In the computation of the functional key, the input circuit is converted to the sequence of polynomials by using RP. These polynomials are converted to the equivalent circuits, then the key generation of pkPAFE computes a tuple of functional keys for these circuits and outputs them as the functional key of a public-key FE. The encryption computes a ciphertext for the input message. For the input message, it is encoded by using RP. Then the encryption of pkPAFE encrypts the encoded message and outputs it as the ciphertext. The decryption is done by combining the decryption of pkPAFE and the decoding algorithm of RP. First, partially decrypted values are computed by using the projective decryption algorithm of pkPAFE. Then, these partially decrypted values are recovered to the circuit value by using the recovering algorithm of pkPAFE and the decoding algorithm of RP.

pkPAFE with RP derives a single-key selective secure public-key functional encryption. By combining the results of [6], it means that pkPAFE and RP can achieve $\mathsf{i}\mathcal{O}$. However, we do not give an instantiation of pkPAFE from other cryptographic primitives as the secret-key case of [3]. It is an important problem to find cryptographic primitives which lead to pkPAFE.

2 Preliminaries

We introduce notions and notations used in this paper. For a finite set X, $x \leftarrow X$ means that x is chosen uniformly at random from X. For any algorithm $\mathcal{A}$, let $y \leftarrow \mathcal{A}(x)$ denote that $\mathcal{A}$ outputs y on input x. When $\mathcal{A}$ is probabilistic, $\mathcal{A}(x)$ denotes a random variable for the output of $\mathcal{A}$ on input x, where the probability is taken over the internal coin flips of $\mathcal{A}$. We allow that $\mathcal{A}$ can output the special symbol $\perp$ to indicate that $\mathcal{A}$ rejects the input and halts with no outputs. We say that a function ε is *negligible* in λ, if for any polynomial p, there exists a natural number λ_0 such that for any $\lambda > \lambda_0$, $\varepsilon(\lambda) < 1/p(\lambda)$.

2.1 Circuits

A circuit is expressed by a directed non-cyclic graph. Nodes in a graph are classified as (normal) gates, input gates or output gates. Input gates are labeled by variables $x_1, \ldots, x_n$, and these are the input for a circuit. In a circuit, at least one gate is set as the output gate, and the output of it is the output of a circuit. Other gates are normal gates and they execute the operation which is set to the gate. Edges in a graph is considered as wires which connect each node in a circuit. When all gates in a circuit execute an *arithmetic* operation, we call it an arithmetic circuit. The size $|C|$ of a circuit C is the number of gates in C.

In this paper, we consider a family of circuits $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$. $\mathcal{C}_\lambda$ is indexed by a natural number λ and $\mathcal{C}_\lambda$ is the set of all circuits $C : \{0,1\}^\lambda \to \{0,1\}^\lambda$. For a circuit $C \in \mathcal{C}_\lambda$, an input for C is a λ-bit string $x \in \{0,1\}^\lambda$, and the output of C on the input x is a λ-bit string $C(x) \in \{0,1\}^\lambda$. We call $C(x)$ the circuit value of C on x. Note that the circuit values are often a single bit, not a string. However, single-bit circuit values are naturally regarded as a subset of multi-bit circuit values. We use a class of circuits of multi-bit circuit values to treat arithmetic circuits as below.

Let q be a prime of λ-bit length and let $\mathbb{F}_q$ be a finite field of characteristic q, respectively. A polynomial p over $\mathbb{F}_q$ means that all coefficients of p are in $\mathbb{F}_q$ and $p(x)$ is in $\mathbb{F}_q$ for any input $x \in \mathbb{F}_q$. We consider a family of arithmetic circuits over $\mathbb{F}_q$. For a family of *arithmetic* circuits $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$, $\mathcal{C}$ is said to be over $\mathbb{F}_q$ if for any $\lambda \in \mathbb{N}$ and $C \in \mathcal{C}_\lambda$, C takes $x \in \mathbb{F}_q$ as a λ-bit string and the circuit value $C(x) \in \{0,1\}^\lambda$ is interpreted as an element of $\mathbb{F}_q$. We use the expression "over $\mathbb{F}_q$" on not only a family $\mathcal{C}$ but also an individual circuit C.

We finally note about the equivalence between polynomials over $\mathbb{F}_q$ and circuits over $\mathbb{F}_q$. We say that a polynomial p over $\mathbb{F}_q$ and an arithmetic circuit C over $\mathbb{F}_q$ are *equivalent* if $p(x) = C(x)$ holds for any $x \in \mathbb{F}_q$. We also note that we can construct an arithmetic circuit $C_p \in \mathcal{C}_\lambda$ over $\mathbb{F}_q$ for a polynomial p over $\mathbb{F}_q$ such that C_p and p are equivalent in polynomial time in λ for any q of bit length λ [3].

2.2 Public-Key Functional Encryption

We consider the public-key functional encryption (FE) for circuits in this paper. We first give a brief description. The public-key functional encryption for a family of circuits $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ consists of four algorithms $\mathsf{FE.Setup}$, $\mathsf{FE.KeyGen}$, $\mathsf{FE.Enc}$ and $\mathsf{FE.Dec}$. $\mathsf{FE.Setup}$ is the setup algorithm. It generates a pair of a public key MPK and a secret key MSK on input security parameter λ. MPK is used to encrypt a message and MSK is used to produce a functional key for the associated circuit, respectively. We note that a security parameter λ is input of form 1^λ to make the input size be λ. $\mathsf{FE.KeyGen}$ is the key generation algorithm. It produces a functional key SK_C for the input circuit $C \in \mathcal{C}_\lambda$ by using the secret key MSK. SK_C is used to decrypt the circuit value $C(x)$ of the message x from the ciphertext CT. $\mathsf{FE.KeyGen}$ is the encryption algorithm. $\mathsf{FE.KeyGen}$ encrypts the input message x to a ciphertext CT by using MPK. $\mathsf{FE.Dec}$ is the decryption algorithm. It computes the circuit value $C(x)$ of the message x from the ciphertext CT by using the functional key SK_C associated the circuit C.

The formal definition of the public-key functional encryption is given as follows.

Definition 1. *Let $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$ be a message space where $\mathcal{X}_\lambda = \{0,1\}^\lambda$, and let $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of circuits, respectively. A public-key functional encryption* $\mathsf{FE} = (\mathsf{FE.Setup}$*,* $\mathsf{FE.KeyGen}$*,*

$\mathsf{FE.Enc}$, $\mathsf{FE.Dec}$) *for* $\mathcal{C}$ *consists of the following four algorithms.*

- $\mathsf{FE.Setup}(1^\lambda)$*: The setup algorithm takes as input a security parameter* 1^λ *and outputs a public key* MPK *and a corresponding secret key* MSK*.*
- $\mathsf{FE.KeyGen}(\mathsf{MSK}, C)$*: The key generation algorithm takes as input the secret key* MSK *and a circuit* $C \in \mathcal{C}_\lambda$*, and outputs a functional key* SK_C*.*
- $\mathsf{FE.Enc}(\mathsf{MPK}, x)$*: The encryption algorithm takes as input the public key* MPK *and a message* $x \in \mathcal{X}_\lambda$*, and outputs a ciphertext* CT*.*
- $\mathsf{FE.Dec}(\mathsf{SK}_C, \mathsf{CT})$*: The decryption algorithm takes as input a functional key* SK_C *and a ciphertext* CT*, and outputs* out*.*

The completeness of the functional encryption is defined as follows.
Completeness : For any $\lambda \in \mathbb{N}$, message $x \in \mathcal{X}_\lambda$ and circuit $C \in \mathcal{C}_\lambda$, it holds that

$$\mathsf{FE.Dec}(\mathsf{FE.KeyGen}(\mathsf{MSK}, C), \mathsf{FE.Enc}(\mathsf{MPK}, x)) = C(x),$$

where $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$.

We consider the selective security on the functional encryption.

Definition 2. *The security game* $\mathsf{Exp}^{\mathsf{sel}}_{\mathcal{A}}(1^\lambda, b)$ *for* $b \in \{0,1\}$ *between the adversary* $\mathcal{A}$ *and the challenger* $\mathcal{I}$ *is defined as follows.*

- **Setup***: The challenger* $\mathcal{I}$ *takes as input a security parameter* 1^λ*.* $\mathcal{I}$ *generates a public key and a secret key* $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$ *and sends* MPK *to* $\mathcal{A}$*.*
- **Challenge***: The adversary* $\mathcal{A}$ *chooses two messages* $(x_0, x_1) \in \mathcal{X}_\lambda$ *and sends them to* $\mathcal{I}$*.* $\mathcal{I}$ *computes* $\mathsf{CT}^* \leftarrow \mathsf{FE.Enc}(\mathsf{MPK}, x_b)$ *and sends it to* $\mathcal{A}$*.*
- **Query***:* $\mathcal{A}$ *chooses a circuit* $C \in \mathcal{C}_\lambda$ *where* $C(x_0) = C(x_1)$ *and sends* C *to* $\mathcal{I}$*.* $\mathcal{I}$ *computes* $\mathsf{SK}_C \leftarrow \mathsf{FE.KeyGen}(\mathsf{MSK}, C)$ *and sends it to* $\mathcal{A}$*.*
- **Output***:* $\mathcal{A}$ *outputs* b' *as the output of* $\mathsf{Exp}^{\mathsf{sel}}_{\mathcal{A}}(1^\lambda, b)$*.*

Note that **Setup** *phase,* **Challenge** *phase and* **Output** *phase are done only one time.* **Query** *phase is repeated at most polynomially many times.*

The advantage $\mathsf{Adv}^{\mathsf{sel}}_{\mathcal{A}}$ *of* $\mathcal{A}$ *on* $\mathsf{Exp}^{\mathsf{sel}}_{\mathcal{A}}(1^\lambda, b)$ *is defined as follows:*

$$\mathsf{Adv}^{\mathsf{sel}}_{\mathcal{A}}(\lambda) = |\mathsf{Pr}[\mathsf{Exp}^{\mathsf{sel}}_{\mathcal{A}}(1^\lambda, 0) = 1] - \mathsf{Pr}[\mathsf{Exp}^{\mathsf{sel}}_{\mathcal{A}}(1^\lambda, 1) = 1]|.$$

We say that FE *is selective secure if for any polynomial-time adversary* $\mathcal{A}$*, there exists a negligible function* $\varepsilon^{\mathsf{sel}}_{\mathcal{A}}$ *such that*

$$\mathsf{Adv}^{\mathsf{sel}}_{\mathcal{A}}(\lambda) \leq \varepsilon^{\mathsf{sel}}_{\mathcal{A}}(\lambda)$$

holds.

A single-key variant of the selective security is defined as follows.

Definition 3. *Consider the* $\mathsf{Exp}^{\mathsf{sel}}_{1,\mathcal{A}}(1^\lambda, b)$*, a variant of* $\mathsf{Exp}^{\mathsf{sel}}_{\mathcal{A}}(1^\lambda, b)$ *in which* **Query** *phase is done at most one time. If* FE *is selective secure under the game* $\mathsf{Exp}^{\mathsf{sel}}_{1,\mathcal{A}}(1^\lambda, b)$*, we call* FE *a single-key selective secure.*

Concerning the efficiency of the functional encryption, we introduce the notion of the sublinearity.

Definition 4. *Let* $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a public-key functional encryption for* $\mathcal{C}$*. We call* FE *sublinear if there exists a polynomial* $p^{\mathsf{SL}}_{\mathsf{FE}}$ *such that the computation time* t_{FE} *of* Enc *satisfies the following formula,*

$$t_{\mathsf{FE}} \leq (l_C)^\varepsilon \cdot p^{\mathsf{SL}}_{\mathsf{FE}}(\lambda),$$

where $0 < \varepsilon < 1$ *and* $l_C = \max_{C \in \mathcal{C}_\lambda}\{|C|\}$*.*

2.3 Randomizing Polynomials Scheme

We introduce the randomizing polynomials scheme (RP) [4][1]. The RP is defined with the target arithmetic circuit family. The RP aims to express a circuit by a sequence of polynomials. We first give a brief description. The randomizing polynomials scheme for a family of arithmetic circuits $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ consists of three algorithms CEncd, $\mathsf{InpEncd}$ and Decd. CEncd is the encoding algorithm for *circuits*. It encodes the input circuit $C \in \mathcal{C}_\lambda$ to the sequence of polynomials $(p_1, \dots, p_N)$. $\mathsf{InpEncd}$ is the encoding algorithm for *messages*. It encodes the input message $x \in \mathcal{X}_\lambda$ to the encoded message $\hat{x}$. Decd the decoding algorithm. It aims to compute the circuit value $C(x)$ from $(p_1, \dots, p_N)$ and $\hat{x}$.

We use the RP as the auxiliary component in the construction of the public-key functional encryption from the public-key projective arithmetic functional encryption which will be described later.

The formal definition of the randomizing polynomials scheme is given as follows.

Definition 5. *Let $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ be the space of messages where $\mathcal{X}_\lambda = \{0,1\}^\lambda$, and $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of arithmetic circuits over the finite field $\mathbb{F}_q$ with a prime q, respectively. Let N and d be fixed naturals and let p_R be a polynomial in λ.*

A randomizing polynomials scheme $\mathsf{RP} = (\mathsf{CEncd}, \mathsf{InpEncd}, \mathsf{Decd})$ over $\mathbb{F}_q$ for $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ consists of the following three deterministic algorithms.

- $\mathsf{CEncd}(1^\lambda, C)$*: The encoder for circuits takes a security parameter 1^λ and a circuit $C \in \mathcal{C}_\lambda$. It outputs the sequence of polynomials $(p_1, \dots, p_N)$ over $\mathbb{F}_q$.*
- $\mathsf{InpEncd}(x, R)$*: The encoder for messages takes a message $x \in \mathcal{X}_\lambda$ and a randomness $R \in \{0,1\}^{l_R}$, where $l_R = p_R(\lambda)$. It outputs the encoded message $\hat{x} \in \mathbb{F}_q$.*
- $\mathsf{Decd}(p_1(\hat{x}), \dots, p_N(\hat{x}))$*: The decoder takes as input the sequence of values $(p_1(\hat{x}), \dots, p_N(\hat{x}))$ and outputs y.*

The completeness and the security of the randomizing polynomials scheme is defined as follows. *Completeness* : For any $\lambda \in \mathbb{N}$, message $x \in \mathcal{X}_\lambda$, circuit $C \in \mathcal{C}_\lambda$ and randomness $R \in \{0,1\}^{l_R}$, it holds that

$$\mathsf{Decd}(p_1(\hat{x}), \cdots, p_N(\hat{x})) = C(x),$$

where $(p_1, \dots, p_N) \leftarrow \mathsf{CEncd}(1^\lambda, C)$ and $\hat{x} \leftarrow \mathsf{InpEncd}(x, R)$.

Definition 6. *The security game $\mathsf{Exp}^{\mathsf{RP}}_{\mathcal{A}}(1^\lambda, b)$ for $b \in \{0,1\}$ between the adversary $\mathcal{A}$ and the challenger $\mathcal{I}$ is defined as follows.*

- **Setup***: The adversary $\mathcal{A}$ chooses a circuit $C \in \mathcal{C}_\lambda$ and a message $x \in \mathcal{X}_\lambda$. $\mathcal{A}$ computes the encoded message $\hat{x} \leftarrow \mathsf{InpEncd}(x, R)$ for $R \in \{0,1\}^{l_R}$. Then $\mathcal{A}$ sends them to the challenger $\mathcal{I}$.*
- **Challenge***: $\mathcal{I}$ computes $(p_1, \dots, p_N) \leftarrow \mathsf{CEncd}(1^\lambda, C)$. Then $\mathcal{I}$ sets $(\theta^b_1, \dots, \theta^b_N)$ according to the input bit b. Namely, $(\theta^b_1, \dots, \theta^b_N)$ is set as*

$$(\theta^0_1, \dots, \theta^0_N) = (p_1(\hat{x}), \dots, p_N(\hat{x})) \qquad \text{when } b = 0,$$
$$(\theta^1_1, \dots, \theta^1_N) = \mathsf{Sim}(1^\lambda, C, C(x)) \qquad \text{when } b = 1,$$

where Sim is a randomized algorithm called the simulator. $\mathcal{I}$ sends $(\theta^b_1, \dots, \theta^b_N)$ to $\mathcal{A}$.

- **Output***: $\mathcal{A}$ outputs b' as the output of $\mathsf{Exp}^{\mathsf{RP}}_{\mathcal{A}}(1^\lambda, b)$.*

The advantage $\mathsf{Adv}^{\mathsf{RP}}_{\mathcal{A}}$ of $\mathcal{A}$ on $\mathsf{Exp}^{\mathsf{RP}}_{\mathcal{A}}(1^\lambda, b)$ is defined as follows.

$$\mathsf{Adv}^{\mathsf{RP}}_{\mathcal{A}}(\lambda) = |\Pr[\mathsf{Exp}^{\mathsf{RP}}_{\mathcal{A}}(1^\lambda, 0) = 1] - \Pr[\mathsf{Exp}^{\mathsf{RP}}_{\mathcal{A}}(1^\lambda, 1) = 1]|.$$

We say that RP *is secure if there exist a negligible function* $\varepsilon^{\mathsf{RP}}$ *and a polynomial-time simulator* Sim *such that for any polynomial-time adversary* $\mathcal{A}$*,*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{RP}}(\lambda) \leq \varepsilon^{\mathsf{RP}}(\lambda)$$

holds.

In this paper, we assume the following two properties on the randomizing polynomials scheme. The first one is concerning the length of encoded messages. The property requires the length $|\hat{x}|$ of the encoded message $\hat{x}$ is sublinear for any message $x \in \mathcal{X}_\lambda$.

Definition 7. *Let* $\mathsf{RP} = (\mathsf{CEncd}, \mathsf{InpEncd}, \mathsf{Decd})$ *be a randomizing polynomials scheme. We call* RP *sublinear if there exists a polynomial* $p_{\mathsf{RP}}^{\mathsf{SL}}$ *such that the length* $|\hat{x}|$ *of the encoded message* $\hat{x}$ *satisfies the following formula for any message* $x \in \mathcal{X}_\lambda$*,*

$$|\hat{x}| \leq |C|^{\varepsilon} \cdot p_{\mathsf{RP}}^{\mathsf{SL}}(\lambda),$$

where $0 < \varepsilon < 1$*.*

The second property is concerning the decoder Decd. B-linearity requires that the decoder Decd can be expressed by a sequence of linear polynomials.

Definition 8. *Let* $B \in \mathbb{F}_q$ *be a constant and let* $(p_1, \ldots, p_N)$ *be the encode of a circuit* C *and* $\hat{x}$ *be the encode of a message* x*. If the decoder* Decd *runs as below for a tuple* $L = (p_1(\hat{x}), \ldots, p_N(\hat{x}))$*, we call* Decd B*-linear:*

1. *initializes the counter* $i = 0$*.*
2. *For* $i = k$*, chooses a linear polynomial* f_k *and computes* $f_k(L) = v_k$*.*
3. *output* v_k *if* $v_k \in \{0, \ldots, B\}$*, or outputs* $\perp$ *if otherwise.*
4. *repeats the steps 2 and 3 until* $i = T$*, outputs* $y = v_T$ *as the output.*

3 Public-Key Projective Arithmetic Functional Encryption

In this section, we define the public-key projective arithmetic functional encryption (pkPAFE). We give a syntax of pkPAFE and definitions of security notions.

We first give a brief description of pkPAFE. The public-key projective arithmetic functional encryption for a family of arithmetic circuits $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ consists of five algorithms pkPAFE.Setup, pkPAFE.KeyGen, pkPAFE.Enc, pkPAFE.ProjectDec and pkPAFE.Recover. The first three algorithms are the same as the corresponding algorithm of the public-key functional encryption. On the other hand, the decryption procedure differs from the public-key FE. In pkPAFE, the decryption is done by the combination of two algorithms pkPAFE.ProjectDec and pkPAFE.Recover. First, the ciphertext CT is decrypted to the by the partial decrypted value ι by the projective decryption algorithm pkPAFE.ProjectDec. Then multiple partial decrypted values are combined to retrieve the linear combination of the circuit values by the recover algorithm pkPAFE.Recover.

The formal definition of public-key projective arithmetic functional encryption is given as follows.

3.1 Definition

Definition 9. *Let* $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$ *be a message space such that* $\mathcal{X}_\lambda \subseteq \mathbb{F}_q$ *with* λ*-bit prime* q*, and let* $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ *be a family of arithmetic circuits over* $\mathbb{F}_q$*, respectively. A public-key projective arithmetic functional encryption* pkPAFE = (pkPAFE.Setup*,* pkPAFE.KeyGen*,* pkPAFE.Enc*,* pkPAFE.ProjectDec*,* pkPAFE.Recover) *for* $\mathcal{C}$ *consists of the following five algorithms.*

- pkPAFE.Setup(1^λ)*: The setup algorithm takes as input a security parameter* 1^λ *and outputs a public key* MPK *and a corresponding secret key* MSK*.*

- $\mathsf{pkPAFE.KeyGen}(\mathsf{MSK}, C)$: *The key generation algorithm takes as input the secret key* MSK *and an arithmetic circuit* $C \in \mathcal{C}_\lambda$, *and outputs a functional key* SK_C.
- $\mathsf{pkPAFE.Enc}(\mathsf{MPK}, x)$: *The encryption algorithm takes as input the public key* MPK *and a message* $x \in \mathcal{X}_\lambda$, *and outputs a ciphertext* CT.
- $\mathsf{pkPAFE.ProjectDec}(\mathsf{SK}_C, \mathsf{CT})$: *The projective decryption algorithm takes as input a functional key* SK_C *and a ciphertext* CT, *and outputs a partial decrypted value* ι.
- $\mathsf{pkPAFE.Recover}(c_1, \iota_1, \ldots, c_\ell, \iota_\ell)$: *The recover algorithm takes as input coefficients* $c_1, \ldots, c_\ell \in \mathbb{F}_q$ *and partial decrypted values* $\iota_1, \ldots, \iota_\ell$, *and outputs* out.

We note that a linear polynomial $f(x_1, ..., x_\ell) = c_1x_1 + \cdots + c_\ell x_\ell$ can be input for $\mathsf{pkPAFE.Recover}$ instead of coefficients $c_1, \ldots, c_\ell$.

The completeness of pkPAFE is defined as follows.

Completeness : For any $\lambda \in \mathbb{N}$, any message $x \in \mathcal{X}_\lambda$ and any arithmetic circuits $C_1, \ldots, C_\ell$, let

- $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{pkPAFE.Setup}(1^\lambda)$,
- $\mathsf{CT} \leftarrow \mathsf{pkPAFE.Enc}(\mathsf{MPK}, x)$,
- $\mathsf{SK}_{C_i} \leftarrow \mathsf{pkPAFE.KeyGen}(\mathsf{MSK}, C_i)$ for all $1 \leq i \leq \ell$,
- $\iota_i \leftarrow \mathsf{pkPAFE.ProjectDec}(\mathsf{SK}_{C_i}, \mathsf{CT})$ for all $1 \leq i \leq \ell$,

where SK_{C_i} denotes the functional key for the arithmetic circuit C_i and ι_i denotes the partial decrypted value of CT with respect to C_i.

Then pkPAFE satisfies the completeness if

$$\mathsf{pkPAFE.Recover}(c_1, \iota_1, \ldots, c_\ell, \iota_\ell) = \sum_{i=1}^{\ell} c_i \cdot C_i(x)$$

holds for any $c_1, \ldots, c_\ell \in \mathbb{F}_q$.

On the efficiency of pkPAFE, we introduce the following notion.

Definition 10. *Let* pkPAFE = ($\mathsf{pkPAFE.Setup}$, $\mathsf{pkPAFE.KeyGen}$, $\mathsf{pkPAFE.Enc}$, $\mathsf{pkPAFE.ProjectDec}$, $\mathsf{pkPAFE.Recover}$) *be a public-key projective arithmetic functional encryption for* $\mathcal{C}$. *We say that* pkPAFE *satisfies the multiplicative overhead in encryption complexity if there exists a polynomial* p_{pkPAFE} *such that the computation time* t_{pkPAFE} *of* $\mathsf{pkPAFE.Enc}$ *satisfies the following formula for any message* $x \in \mathcal{X}_\lambda$,

$$t_{\mathsf{pkPAFE}} \leq |x| \cdot p_{\mathsf{pkPAFE}}(\lambda).$$

3.2 Security Notions

In order to define the security of pkPAFE, we introduce the following two algorithms.

- $\mathsf{sfKG}(\mathsf{MSK}, C, \theta)$: The semi-functional key generation algorithm takes as input the secret key MSK, a circuit $C \in \mathcal{C}_\lambda$ and a value θ associated with C, and outputs a semi-functional key sfSK_C.
- $\mathsf{sfEnc}(\mathsf{MPK}, 1^{\ell_{\mathsf{inp}}})$: The semi-functional encryption algorithm takes as input the public key MPK and the size of input $1^{\ell_{\mathsf{inp}}}$, and outputs a semi-functional ciphertext sfCT.

We consider two notions of indistinguishability concerning functional keys and ciphertexts as the security notions for pkPAFE. The formal definitions are as follows.

Definition 11. *The security game* $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{KeyInd}}(1^\lambda, b)$ *for* $b \in \{0,1\}$ *between the adversary* $\mathcal{A}$ *and the challenger* $\mathcal{I}$ *is defined as follows.*

- **Setup**: *The challenger* $\mathcal{I}$ *takes as input a security parameter* 1^λ. $\mathcal{I}$ *generates a public key and a secret key* $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{pkPAFE.Setup}(1^\lambda)$ *and sends* MPK *to* $\mathcal{A}$.
- **Adversary's Query**: *The adversary* $\mathcal{A}$ *queries the following items:*
  - *a tuple of circuits and associated values* $(C_1^0, \theta_1, \ldots, C_\ell^0, \theta_\ell)$ *where* $\theta_j \in \mathbb{F}_q$ *for all* $1 \leq j \leq \ell$.
  - *a tuple of circuits* $(C_1^1, \ldots, C_{\ell'}^1)$.
  - *a challenge circuit and an associated value* (C^*, θ^*).
- **Challenger's Response**: *The challenger* $\mathcal{I}$ *computes the following items for all* j*:*
  - $\mathsf{SK}_{C_j^0} \leftarrow \mathsf{sfKG}(\mathsf{MSK}, C_j^0, \theta_j)$,
  - $\mathsf{SK}_{C_j^1} \leftarrow \mathsf{pkPAFE.KeyGen}(\mathsf{MSK}, C_j^1)$.

  $\mathcal{I}$ *also computes* SK_{C^*} *according to the input bit* b *as follows:*

  $$\mathsf{SK}_{C^*} \leftarrow \mathsf{sfKG}(\mathsf{MSK}, C^*, \theta^*) \qquad \text{when } b = 0,$$
  $$\mathsf{SK}_{C^*} \leftarrow \mathsf{pkPAFE.KeyGen}(\mathsf{MSK}, C^*) \qquad \text{when } b = 1.$$

  Then $\mathcal{I}$ *returns* $((\mathsf{SK}_{C_1^0}, \ldots, \mathsf{SK}_{C_\ell^0}), (\mathsf{SK}_{C_1^1}, \ldots, \mathsf{SK}_{C_\ell^1}), \mathsf{SK}_{C^*})$.
- **Output**: $\mathcal{A}$ *outputs* b' *as the output of* $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{KeyInd}}(1^\lambda, b)$.

The advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{KeyInd}}$ *of* $\mathcal{A}$ *on* $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{KeyInd}}(1^\lambda, b)$ *is defined as follows.*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{KeyInd}}(\lambda) = |\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{KeyInd}}(1^\lambda, 0) = 1] - \Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{KeyInd}}(1^\lambda, 1) = 1]|.$$

We say that pkPAFE *has the indistinguishability of semi-functional keys if there exists a negligible function* $\varepsilon^{\mathsf{KeyInd}}$ *such that for any polynomial-time adversary* $\mathcal{A}$,

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{KeyInd}}(\lambda) \leq \varepsilon^{\mathsf{KeyInd}}(\lambda)$$

holds

Definition 12. *The security game* $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{CTInd}}(1^\lambda, b)$ *for* $b \in \{0,1\}$ *between the adversary* $\mathcal{A}$ *and the challenger* $\mathcal{I}$ *is defined as follows.*

- **Setup**: *The challenger* $\mathcal{I}$ *takes as input a security parameter* 1^λ. $\mathcal{I}$ *generates a public key and a secret key* $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{pkPAFE.Setup}(1^\lambda)$ *and sends* MPK *to* $\mathcal{A}$.
- **Challenge**: *The adversary* $\mathcal{A}$ *sends a challenge message* x^* *to the challenger. Then the challenger* $\mathcal{I}$ *returns* CT^* *according to the input bit* b *as follows:*

  $$\mathsf{CT}^* \leftarrow \mathsf{sfEnc}(\mathsf{MPK}, 1^{|x^*|}) \qquad \text{when } b = 0,$$
  $$\mathsf{CT}^* \leftarrow \mathsf{pkPAFE.Enc}(\mathsf{MPK}, x^*) \qquad \text{when } b = 1.$$
- **Adversary's Query**: *The adversary queries a tuple of circuits and associated values* $(C_1, \theta_1, \ldots, C_\ell, \theta_\ell)$ *where* $\theta_j = C_j(x^*)$ *for all* $1 \leq j \leq \ell$.
- **Challenger's Response**: *The challenger* $\mathcal{I}$ *returns* $\mathsf{SK}_{C_j} \leftarrow \mathsf{sfKG}(\mathsf{MSK}, C_j, \theta_j)$ *for all* $1 \leq j \leq \ell$.
- **Output**: $\mathcal{A}$ *outputs* b' *as the output of* $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{CTInd}}(1^\lambda, b)$.

The advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{CTInd}}$ *of* $\mathcal{A}$ *on* $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{CTInd}}(1^\lambda, b)$ *is defined as follows.*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{CTInd}}(\lambda) = |\mathsf{Pr}[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{CTInd}}(1^\lambda, 0) = 1] - \mathsf{Pr}[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{CTInd}}(1^\lambda, 1) = 1]|.$$

We say that pkPAFE *has the indistinguishability of semi-functional ciphertexts if there exists a negligible function* $\varepsilon^{\mathsf{CTInd}}$ *such that for any polynomial-time adversary* $\mathcal{A}$*,*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{CTInd}}(\lambda) \leq \varepsilon^{\mathsf{CTInd}}(\lambda)$$

holds

4 Single-Key Selective Secure Functional Encryption from pkPAFE

We show that a single-key selective secure sublinear functional encryption can be derived from a public-key projective arithmetic functional encryption. Since it is known that the indistinguishability obfuscation can be constructed from a single-key selective secure sublinear functional encryption, our result means that an $\mathsf{i}\mathcal{O}$ is constructed from a public-key projective arithmetic functional encryption.

We give a construction of a functional encryption from pkPAFE and then prove that the resulting functional encryption is single-key selective secure and sublinear.

4.1 Construction

Let $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ be a message space such that $\mathcal{X}_\lambda \subseteq \mathbb{F}_q$ with λ-bit prime q, and let $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of arithmetic circuits over $\mathbb{F}_q$, respectively. Let $\mathsf{pkPAFE} = (\mathsf{pkPAFE.Setup}, \mathsf{pkPAFE.KeyGen}, \mathsf{pkPAFE.Enc}, \mathsf{pkPAFE.ProjectDec}, \mathsf{pkPAFE.Recover})$ be a public-key projective arithmetic functional encryption for $\mathcal{C}$, and let $\mathsf{RP} = (\mathsf{CEncd}, \mathsf{InpEncd}, \mathsf{Decd})$ be a randomizing polynomials scheme such that Decd is 0-linear.

The functional encryption $\mathsf{FE}_{\mathsf{ours}} = (\mathsf{FE}_{\mathsf{ours}}.\mathsf{Setup}, \mathsf{FE}_{\mathsf{ours}}.\mathsf{KeyGen}, \mathsf{FE}_{\mathsf{ours}}.\mathsf{Enc}, \mathsf{FE}_{\mathsf{ours}}.\mathsf{Dec})$ for $\mathcal{C}$ is constructed as follows.

- $\mathsf{FE}_{\mathsf{ours}}.\mathsf{Setup}(1^\lambda)$: The setup algorithm takes as input a security parameter 1^λ and computes $\mathsf{pkPAFE.Setup}(1^\lambda) \to (\mathsf{MPK}, \mathsf{MSK})$. Then $\mathsf{FE}_{\mathsf{ours}}.\mathsf{Setup}$ outputs a public key MPK and a secret key MSK.

- $\mathsf{FE}_{\mathsf{ours}}.\mathsf{KeyGen}(\mathsf{MSK}, C)$: The key generation algorithm takes as input the secret key MSK and a circuit $C \in \mathcal{C}_\lambda$. $\mathsf{FE}_{\mathsf{ours}}.\mathsf{KeyGen}$ runs as follows:

  1. computes $\mathsf{CEncd}(1^\lambda, C) \to (p_1, \dots, p_N)$.
  2. constructs an arithmetic circuit C_{p_i} which is equivalent to the polynomial p_i for all $1 \leq i \leq N$.
  3. computes $\mathsf{pkPAFE.KeyGen}(\mathsf{MSK}, C_{p_i}) \to \mathsf{SK}_{C_{p_i}}$ for all $1 \leq i \leq N$.
  4. outputs a functional key $\mathsf{SK}_C = (\mathsf{SK}_{C_{p_1}}, \dots, \mathsf{SK}_{C_{p_N}})$.

- $\mathsf{FE}_{\mathsf{ours}}.\mathsf{Enc}(\mathsf{MPK}, x)$: The encryption algorithm takes as input the public key MPK and a message $x \in \mathcal{X}_\lambda$. $\mathsf{FE}_{\mathsf{ours}}.\mathsf{Enc}$ runs as follows:

  1. chooses a randomness $R \leftarrow \{0,1\}^{l_R}$ uniformly at random.
  2. computes $\mathsf{InpEncd}(x, R) \to \hat{x}$.
  3. computes $\mathsf{pkPAFE.Enc}(\mathsf{MPK}, \hat{x}) \to \mathsf{CT}$.
  4. outputs a ciphertext CT.

- $\mathsf{FE_{ours}}.\mathsf{Dec}(\mathsf{SK}_C, \mathsf{CT})$: The decryption algorithm takes as input a functional key SK_C and a ciphertext CT. $\mathsf{FE_{ours}}.\mathsf{Dec}$ runs as follows:
  1. computes $\mathsf{pkPAFE.ProjectDec}(\mathsf{SK}_{C_{p_i}}, \mathsf{CT}) \to \iota_i$, for all $1 \le i \le N$.
  2. initializes a counter $i = 0$.
  3. For $i = k$, chooses a linear polynomial f_k by using Decd.
  4. computes $\mathsf{pkPAFE.Recover}(f_k, \iota_1, \ldots, \iota_N) \to \mathsf{out}_k$.
  5. repeats the steps 3 and 4 until $i = T$.
  6. outputs $\mathsf{out} = \mathsf{out}_T$.

The proposed scheme $\mathsf{FE_{ours}}$ satisfies the completeness and is sublinear by the following theorems.

Theorem 1. *Assume that* pkPAFE *satisfies the completeness and* RP *satisfies the completeness. Let* Decd *be* 0*-linear. Then* $\mathsf{FE_{ours}}$ *satisfies the completeness.*

Proof. Let λ be a security parameter, $x \in \mathcal{X}_\lambda$ be a message and $C \in \mathcal{C}_\lambda$ be a circuit. Then, the functional SK_C is $\mathsf{SK}_C = (\mathsf{SK}_{C_{p_1}}, \ldots, \mathsf{SK}_{C_{p_N}})$, where $\mathsf{SK}_{C_{p_i}} \leftarrow \mathsf{pkPAFE.KeyGen}(\mathsf{MSK}, C_{p_i})$ and C_{p_i} is a circuit which is equivalent to the polynomial p_i for $(p_1, \ldots, p_N) \leftarrow \mathsf{CEncd}(1^\lambda, C)$. Namely, it holds that $C_{p_i}(x) = p_i(x)$ for any $x \in \mathcal{X}_\lambda$ and all $1 \le i \le N$. The ciphertext CT is $\mathsf{CT} \leftarrow \mathsf{pkPAFE.Enc}(\mathsf{MPK}, \hat{x})$ where $\hat{x} \leftarrow \mathsf{InpEncd}(x, R)$ for a randomness $R \in \{0,1\}^{l_R}$.

Let SK_C and CT be the input for $\mathsf{FE_{ours}}.\mathsf{Dec}$. The decryption algorithm computes $\mathsf{pkPAFE.ProjectDec}(\mathsf{SK}_{C_{p_i}}, \mathsf{CT}) \to \iota_i$, for all $1 \le i \le N$, where $\iota_1, \ldots, \iota_N$ are partial decrypted values. For the counter $i = k$, $\mathsf{FE_{ours}}.\mathsf{Dec}$ computes $\mathsf{pkPAFE.Recover}(f_k, \iota_1, \ldots, \iota_N) \to \mathsf{out}_k$ with a linear polynomial f_k which is chosen by the 0-linear decoder Decd. Then $\mathsf{out}_k = f_k(C_{p_1}(\hat{x}), \ldots, C_{p_N}(\hat{x})) = f_k(p_1(\hat{x}), \ldots, p_N(\hat{x})) = v_k$ follows form the completeness of pkPAFE. Since RP is complete, $\mathsf{out} = \mathsf{out}_T = v_T = \mathsf{Decd}(p_1(\hat{x}), \cdots, p_N(\hat{x})) = C(x)$ follows. □

Theorem 2. *Assume that* pkPAFE *satisfies the multiplicative overhead in encryption complexity and* RP *is sublinear. Then* $\mathsf{FE_{ours}}$ *is sublinear.*

Proof. We evaluate the computation time $t_{\mathsf{FE_{ours}}}$ of the encryption algorithm $\mathsf{FE_{ours}}.\mathsf{Enc}$ for input x. Since pkPAFE satisfies the multiplicative overhead in encryption complexity, we have

$$t_{\mathsf{FE_{ours}}} \le |\hat{x}| \cdot p_{\mathsf{pkPAFE}}(\lambda)$$

for the polynomial p_{pkPAFE}.

For $|\hat{x}|$, the following holds from the sublinearity of RP,

$$|\hat{x}| \le |C|^\varepsilon \cdot p^{\mathsf{SL}}_{\mathsf{RP}}(\lambda),$$

for $0 < \varepsilon < 1$ with the polynomial $p^{\mathsf{SL}}_{\mathsf{RP}}$.

By $\mathcal{X}_\lambda \subseteq \mathbb{F}_q$, we have

$$\begin{aligned} t_{\mathsf{FE_{ours}}} &\le |\hat{x}| \cdot p_{\mathsf{pkPAFE}}(\lambda) \\ &\le |C|^\varepsilon \cdot p^{\mathsf{SL}}_{\mathsf{RP}}(\lambda) \cdot p_{\mathsf{pkPAFE}}(\lambda) \\ &\le |C|^\varepsilon \cdot p^*(\lambda) \end{aligned}$$

for the polynomial $p^* = p^{\mathsf{SL}}_{\mathsf{RP}} \cdot p_{\mathsf{pkPAFE}}$. Then the statement follows. □

4.2 Security

We show that our proposed scheme is single-key selective secure.

Theorem 3. *Assume that* pkPAFE *has the indistinguishability of semi-functional keys and the indistinguishability of semi-functional ciphertexts, and* RP *is secure. Then* $\mathsf{FE_{ours}}$ *is single-key selective secure.*

Proof. We prove the theorem by the hybrid argument. For the games Hyb_k $(1 \le k \le 6)$ except $k = 2$ and 5, we denote by Succ_k the event that the adversary outputs 1 in Hyb_k. For $k = 2$ and 5, $\mathsf{Succ}_{k,i}$ $(1 \le i \le N+1)$ denotes the event that the adversary outputs 1 in $\mathsf{Hyb}_{2,i}$ and $\mathsf{Hyb}_{5,i}$.

Hyb_1: The description of the game is as follows.

- **Setup**: The challenger $\mathcal{I}$ runs as follows.
  1. computes $\mathsf{FE_{ours}.Setup}(1^\lambda) \to (\mathsf{MPK}, \mathsf{MSK})$.
  2. sends MPK to the adversary $\mathcal{A}$.
- **Challenge**: $\mathcal{A}$ chooses two messages $(x_0, x_1) \in \mathcal{X}_\lambda$ and sends them to $\mathcal{I}$. $\mathcal{I}$ computes $\mathsf{CT} \leftarrow \mathsf{FE_{ours}.Enc}(\mathsf{MPK}, x_0)$ and sends CT to $\mathcal{A}$.
- **Query**: $\mathcal{A}$ chooses a circuit $C \in \mathcal{C}_\lambda$ where $C(x_0) = C(x_1)$ and sends C to $\mathcal{I}$. $\mathcal{I}$ computes $\mathsf{SK}_C \leftarrow \mathsf{FE_{ours}.KeyGen}(\mathsf{MSK}, C)$ and sends SK_C to $\mathcal{A}$.
- **Output**: $\mathcal{A}$ outputs b.

For this game Hyb_1, the following lemma holds.

Lemma 1.

$$\Pr[\mathsf{Succ}_1] = \Pr[\mathsf{Exp}^{\mathsf{sel}}_{1,\mathcal{A}}(1^\lambda, 0) = 1]. \tag{1}$$

Proof. By Definition 2, Definition 3 and the description of Hyb_1, Hyb_1 coincides with $\mathsf{Exp}^{\mathsf{sel}}_{1,\mathcal{A}}(1^\lambda, 0)$. Therefore $\Pr[\mathsf{Succ}_1] = \Pr[\mathsf{Exp}^{\mathsf{sel}}_{1,\mathcal{A}}(1^\lambda, 0) = 1]$ follows. □

$\mathsf{Hyb}_{2,i}$ $(1 \le i \le N+1)$: $\mathsf{Hyb}_{2,i}$ proceeds in the same way as Hyb_1 except **Query** phase. **Query** phase is changed as follows.

Query phase: $\mathcal{A}$ chooses a circuit $C \in \mathcal{C}_\lambda$ where $C(x_0) = C(x_1)$ and sends C to $\mathcal{I}$. $\mathcal{I}$ computes SK_C as following steps:

1. computes $(p_1, ..., p_N) \leftarrow \mathsf{CEncd}(1^\lambda, C)$.
2. constructs an arithmetic circuits C_{p_j} which is equivalent to the polynomial p_j for all $1 \le j \le N$.
3. sets $\theta_j = p_j(\hat{x_0})$ for all $1 \le j \le N$, where $\hat{x_0}$ is the one computed during $\mathsf{FE_{ours}.Enc}(\mathsf{MPK}, x_0)$ in **Challenge** phase.
4. sets $\begin{cases} \mathsf{sfSK}_{C_{p_j}} \leftarrow \mathsf{sfKG}(\mathsf{MSK}, C_{p_j}, \theta_j) & \text{for } j < i \\ \mathsf{SK}_{C_{p_j}} \leftarrow \mathsf{PAFE.KeyGen}(\mathsf{MSK}, C_{p_j}) & \text{for } j \ge i \end{cases}$
5. sets $\mathsf{SK}_C = (\mathsf{sfSK}_{C_{p_1}}, \dots, \mathsf{sfSK}_{C_{p_{i-1}}}, \mathsf{SK}_{C_{p_i}}, \dots, \mathsf{SK}_{C_{p_N}})$.
6. outputs SK_C.

For the difference between Hyb_1 and $\mathsf{Hyb}_{2,1}$, we have the following lemma.

Lemma 2.

$$\Pr[\mathsf{Succ}_{2,1}] = \Pr[\mathsf{Succ}_1]. \tag{2}$$

Proof. The difference between Hyb_1 and $\mathsf{Hyb}_{2,1}$ is **Query** phase which outputs SK_C on input a circuit C. In $\mathsf{Hyb}_{2,1}$, all $\mathsf{SK}_{C_{p_j}}$ $(1 \le j \le N)$ is computed as $\mathsf{SK}_{C_{p_j}} \leftarrow \mathsf{PAFE.KeyGen}(\mathsf{MSK}, C_{p_j})$. Then, $\mathsf{SK}_C = (\mathsf{SK}_{C_{p_1}}, \dots, \mathsf{SK}_{C_{p_N}})$ follows. This means that $\mathsf{Hyb}_{2,1}$ coincides with Hyb_1 because $\mathsf{SK}_C \leftarrow \mathsf{FE_{ours}.KeyGen}(\mathsf{MSK}, C)$ in Hyb_1. Therefore $\Pr[\mathsf{Succ}_{2,1}] = \Pr[\mathsf{Succ}_1]$ follows. □

The indistinguishability of semi-functional keys on pkPAFE implies that the difference among $\mathsf{Hyb}_{2,i}$ is negligible.

Lemma 3. *Assume that* pkPAFE *has the indistinguishability of semi-functional keys. Then for all* $1 \leq i \leq N$*,*

$$|\mathsf{Pr}[\mathsf{Succ}_{2,i+1}] - \mathsf{Pr}[\mathsf{Succ}_{2,i}]| \leq \varepsilon^{\mathsf{KeyInd}}(\lambda). \tag{3}$$

Proof. Fix an index i ($1 \leq i \leq N$). We construct an adversary $\mathcal{B}_2$ against the indistinguishability of semi-functional keys of pkPAFE. Let $\mathcal{A}$ be an adversary on the games $\mathsf{Hyb}_{2,i}$ and $\mathsf{Hyb}_{2,i+1}$. $\mathcal{B}_2$ plays a role of the adversary in the game $\mathsf{Exp}^{\mathsf{KeyInd}}_{\mathcal{B}_2}(1^\lambda, b)$, and a role of the challenger in the games $\mathsf{Hyb}_{2,i}$ and $\mathsf{Hyb}_{2,i+1}$. The description of $\mathcal{B}_2$ is as follows.

Setup: $\mathcal{B}_2$ takes an input MPK from the challenger $\mathcal{I}$ of $\mathsf{Exp}^{\mathsf{KeyInd}}_{\mathcal{B}_2}(1^\lambda, b)$ then sends MPK to $\mathcal{A}$ as the input.

Challenge: For messages (x_0, x_1) from $\mathcal{A}$, $\mathcal{B}_2$ computes $\mathsf{CT} \leftarrow \mathsf{FE}_{\mathsf{ours}}.\mathsf{Enc}(\mathsf{MPK}, x_0)$ and sends CT to $\mathcal{A}$.

Query: For a circuit C from $\mathcal{A}$, $\mathcal{B}_2$ computes SK_C as following steps:

1. computes $(p_1, ..., p_N) \leftarrow \mathsf{CEncd}(1^\lambda, C)$.
2. constructs an arithmetic circuits C_{p_j} which is equivalent to the polynomial p_j for all $1 \leq j \leq N$.
3. sets $\theta_j = p_j(\hat{x_0})$ for all $1 \leq j \leq N$, where $\hat{x_0}$ is the one computed during $\mathsf{FE}_{\mathsf{ours}}.\mathsf{Enc}(\mathsf{MPK}, x_0)$ in **Challenge** phase.
4. sends the following items to the challenger $\mathcal{I}$:
   - circuits and corresponding associated values $(C_{p_1}, \theta_1, \ldots, C_{p_{i-1}}, \theta_{i-1})$,
   - circuits $(C_{p_{i+1}}, \ldots, C_{p_N})$, and
   - the challenge circuit and its associated value (C_{p_i}, θ_i).
5. receives SK_C from $\mathcal{I}$, then sends SK_C to $\mathcal{A}$.

Output: For the output b' of $\mathcal{A}$, $\mathcal{B}_2$ outputs b'.

For SK_C computed by $\mathcal{B}_2$, it follows from Definition 11 that

- $\mathsf{SK}_{C_{p_k}} = \mathsf{sfSK}_{C_{p_k}} \leftarrow \mathsf{sfKG}(\mathsf{MSK}, C_{p_k}, \theta_k)$ for $1 \leq k \leq i-1$,
- $\mathsf{SK}_{C_{p_k}} \leftarrow \mathsf{pkPAFE.KeyGen}(\mathsf{MSK}, C_{p_k})$ for $i+1 \leq k \leq N$,
- $\begin{cases} \mathsf{SK}_{C_{p_i}} = \mathsf{sfSK}_{C_{p_i}} \leftarrow \mathsf{sfKG}(\mathsf{MSK}, C_{p_i}, \theta_i) & \text{if } b = 0 \\ \mathsf{SK}_{C_{p_i}} \leftarrow \mathsf{pkPAFE.KeyGen}(\mathsf{MSK}, C_{p_i}) & \text{if } b = 1 \end{cases}.$

Namely, $\mathsf{SK}_C = (\mathsf{sfSK}_{C_{p_1}}, \ldots, \mathsf{sfSK}_{C_{p_i}}, \mathsf{SK}_{C_{p_{i+1}}}, \ldots, \mathsf{SK}_{C_{p_N}})$ if $b = 0$,
or $\mathsf{SK}_C = (\mathsf{sfSK}_{C_{p_1}}, \ldots, \mathsf{sfSK}_{C_{p_{i-1}}}, \mathsf{SK}_{C_{p_i}}, \ldots, \mathsf{SK}_{C_{p_N}})$ otherwise. Therefore $\mathcal{B}_2$ coincides with the challenger of $\mathsf{Hyb}_{2,i+1}$ when $b = 0$, and coincides with the challenger of $\mathsf{Hyb}_{2,i}$ when $b = 1$.

Then we have

$$\begin{aligned} &\mathsf{Pr}[\mathsf{Exp}^{\mathsf{KeyInd}}_{\mathcal{B}_2}(1^\lambda, 0) = 1] \\ &= \mathsf{Pr}\left[b' = 1 \mid \mathsf{SK}_C = (\mathsf{sfSK}_{C_{p_1}}, \ldots, \mathsf{sfSK}_{C_{p_i}}, \mathsf{SK}_{C_{p_{i+1}}}, \ldots, \mathsf{SK}_{C_{p_N}})\right] \\ &= \mathsf{Pr}[\mathsf{Succ}_{2,i+1}], \end{aligned}$$

and

$$\begin{aligned} &\mathsf{Pr}[\mathsf{Exp}^{\mathsf{KeyInd}}_{\mathcal{B}_2}(1^\lambda, 1) = 1] \\ &= \mathsf{Pr}\left[b' = 1 \mid \mathsf{SK}_C = (\mathsf{sfSK}_{C_{p_1}}, \ldots, \mathsf{sfSK}_{C_{p_{i-1}}}, \mathsf{SK}_{C_{p_i}}, \ldots, \mathsf{SK}_{C_{p_N}})\right] \\ &= \mathsf{Pr}[\mathsf{Succ}_{2,i}]. \end{aligned}$$

The advantage $\mathsf{Adv}^{\mathsf{KeyInd}}_{\mathcal{B}_2}(\lambda)$ of $\mathcal{B}_2$ on the indistinguishability of semi-functional keys is computed by

$$\begin{aligned}\mathsf{Adv}^{\mathsf{KeyInd}}_{\mathcal{B}_2}(\lambda) &= |\mathsf{Pr}[\mathsf{Exp}^{\mathsf{KeyInd}}_{\mathcal{B}_2}(1^\lambda, 0) = 1] - \mathsf{Pr}[\mathsf{Exp}^{\mathsf{KeyInd}}_{\mathcal{B}_2}(1^\lambda, 1) = 1]| \\ &= |\mathsf{Pr}[\mathsf{Succ}_{2,i+1}] - \mathsf{Pr}[\mathsf{Succ}_{2,i}]|.\end{aligned}$$

By the assumption on the statement that pkPAFE has the indistinguishability of semi-functional keys, $\mathsf{Adv}^{\mathsf{KeyInd}}_{\mathcal{B}_2}(\lambda) \leq \varepsilon^{\mathsf{KeyInd}}(\lambda)$ follows. Thus we have

$$|\mathsf{Pr}[\mathsf{Succ}_{2,i+1}] - \mathsf{Pr}[\mathsf{Succ}_{2,i}]| \leq \varepsilon^{\mathsf{KeyInd}}(\lambda).$$

□

Hyb_3: Hyb_3 proceeds in the same way as $\mathsf{Hyb}_{2,N+1}$ except **Challenge** phase. The description of **Challenge** phase in Hyb_3 is as follows.

Challenge phase: $\mathcal{A}$ chooses two messages $(x_0, x_1) \in \mathcal{X}_\lambda$ and sends them to $\mathcal{I}$. $\mathcal{I}$ computes $\hat{x_0} \leftarrow \mathsf{InpEncd}(x_0, R)$ for a randomness $R \leftarrow \{0,1\}^{l_R}$. Then $\mathcal{I}$ computes $\mathsf{CT} \leftarrow \mathsf{sfEnc}(\mathsf{MPK}, 1^{|\hat{x_0}|})$ and sends CT to $\mathcal{A}$.

The indistinguishability of semi-functional ciphertexts on pkPAFE implies that the difference between $\mathsf{Hyb}_{2,N+1}$ and Hyb_3 is negligible.

Lemma 4. *Assume that* pkPAFE *has the indistinguishability of semi-functional ciphertexts. Then*

$$|\mathsf{Pr}[\mathsf{Succ}_3] - \mathsf{Pr}[\mathsf{Succ}_{2,N+1}]| \leq \varepsilon^{\mathsf{CTInd}}(\lambda). \tag{4}$$

Proof. We construct an adversary $\mathcal{B}_{23}$ against the indistinguishability of semi-functional ciphertexts of pkPAFE. Let $\mathcal{A}$ be an adversary on the games $\mathsf{Hyb}_{2,N+1}$ and Hyb_3. $\mathcal{B}_{23}$ plays a role of the adversary in the game $\mathsf{Exp}^{\mathsf{CTInd}}_{\mathcal{B}_{23}}(1^\lambda, b)$, and a role of the challenger in the games $\mathsf{Hyb}_{2,N+1}$ and Hyb_3. The description of $\mathcal{B}_{23}$ is as follows.

Setup: $\mathcal{B}_{23}$ takes an input MPK from the challenger $\mathcal{I}$ of $\mathsf{Exp}^{\mathsf{CTInd}}_{\mathcal{B}_{23}}(1^\lambda, b)$ then sends MPK to $\mathcal{A}$ as the input.

Challenge: For messages (x_0, x_1) from $\mathcal{A}$, $\mathcal{B}_{23}$ computes $\hat{x_0} \leftarrow \mathsf{InpEncd}(x_0, R)$ for a randomness $R \leftarrow \{0,1\}^{l_R}$. $\mathcal{B}_{23}$ sends $\hat{x_0}$ to the challenger $\mathcal{I}$. After $\mathcal{B}_{23}$ receives CT from $\mathcal{I}$, $\mathcal{B}_{23}$ returns CT to $\mathcal{A}$.

Query: For a circuit C from $\mathcal{A}$, $\mathcal{B}_{23}$ computes SK_C as following steps:

1. computes $(p_1, ..., p_N) \leftarrow \mathsf{CEncd}(1^\lambda, C)$.
2. constructs an arithmetic circuits C_{p_j} which is equivalent to the polynomial p_j for all $1 \leq j \leq N$.
3. sets $\theta_j = p_j(\hat{x_0})$ for all $1 \leq j \leq N$, where $\hat{x_0}$ is the one computed in **Challenge** phase.
4. sends $(C_{p_1}, \theta_1, \ldots, C_{p_N}, \theta_N)$ to $\mathcal{I}$.
5. receives SK_C from $\mathcal{I}$, then sends SK_C to $\mathcal{A}$.

Output: For the output b' of $\mathcal{A}$, $\mathcal{B}_{23}$ outputs b'.

For SK_C computed by $\mathcal{B}_{23}$, it follows from Definition 12 that $\mathsf{SK}_C = (\mathsf{sfSK}_{C_{p_1}}, \ldots, \mathsf{sfSK}_{C_{p_N}})$ for $\mathsf{sfSK}_{C_{p_j}} \leftarrow \mathsf{sfKG}(\mathsf{MSK}, C_{p_j}, \theta_j)$ $(1 \leq j \leq N)$. This coincides with the computation of SK_C in $\mathsf{Hyb}_{2,N+1}$.

For CT, CT is computed by $\mathsf{CT} = \mathsf{sfCT} \leftarrow \mathsf{sfEnc}(\mathsf{MPK}, 1^{|\hat{x_0}|})$ if $b = 0$, or $\mathsf{CT} \leftarrow \mathsf{pkPAFE.Enc}(\mathsf{MPK}, \hat{x_0})$ if $b = 1$ from Definition 12. Therefore $\mathcal{B}_{23}$ coincides the challenger of Hyb_3 when $b = 0$, and coincides the challenger of $\mathsf{Hyb}_{2,N+1}$ when $b = 1$.

Then we have

$$\begin{aligned}&\mathsf{Pr}[\mathsf{Exp}^{\mathsf{CTInd}}_{\mathcal{B}_{23}}(1^\lambda,0)=1]\\&=\mathsf{Pr}\left[b'=1\mid \mathsf{CT}=\mathsf{sfCT}\leftarrow\mathsf{sfEnc}(\mathsf{MPK},1^{|\hat{x_0}|})\right]\\&=\mathsf{Pr}[\mathsf{Succ}_3],\end{aligned}$$

and

$$\begin{aligned}&\mathsf{Pr}[\mathsf{Exp}^{\mathsf{CTInd}}_{\mathcal{B}_{23}}(1^\lambda,1)=1]\\&=\mathsf{Pr}\left[b'=1\mid \mathsf{CT}\leftarrow\mathsf{pkPAFE.Enc}(\mathsf{MPK},\hat{x_0})\right]\\&=\mathsf{Pr}[\mathsf{Succ}_{2,N+1}].\end{aligned}$$

The advantage $\mathsf{Adv}^{\mathsf{CTInd}}_{\mathcal{B}_{23}}(\lambda)$ of $\mathcal{B}_{23}$ on the indistinguishability of semi-functional ciphertexts is computed by

$$\begin{aligned}\mathsf{Adv}^{\mathsf{CTInd}}_{\mathcal{B}_{23}}(\lambda)&=|\mathsf{Pr}[\mathsf{Exp}^{\mathsf{CTInd}}_{\mathcal{B}_{23}}(1^\lambda,0)=1]-\mathsf{Pr}[\mathsf{Exp}^{\mathsf{CTInd}}_{\mathcal{B}_{23}}(1^\lambda,1)=1]|\\&=|\mathsf{Pr}[\mathsf{Succ}_3]-\mathsf{Pr}[\mathsf{Succ}_{2,N+1}]|.\end{aligned}$$

By the assumption on the statement that pkPAFE has the indistinguishability of semi-functional ciphertexts, $\mathsf{Adv}^{\mathsf{CTInd}}_{\mathcal{B}_{23}}(\lambda)\le\varepsilon^{\mathsf{CTInd}}(\lambda)$. Thus we have

$$|\mathsf{Pr}[\mathsf{Succ}_3]-\mathsf{Pr}[\mathsf{Succ}_{2,N+1}]|\le\varepsilon^{\mathsf{CTInd}}(\lambda).$$

□

Hyb_4: Hyb_4 proceeds in the same way as Hyb_3 except **Challenge** phase **Query** phase. The description of **Challenge** phase and **Query** phase in Hyb_4 is as follows.

Challenge phase: $\mathcal{A}$ chooses two messages $(x_0,x_1)\in\mathcal{X}_\lambda$ and sends them to $\mathcal{I}$. $\mathcal{I}$ computes $\hat{x_1}\leftarrow\mathsf{InpEncd}(x_1,R)$ for a randomness $R\leftarrow\{0,1\}^{l_R}$. Then $\mathcal{I}$ computes $\mathsf{CT}\leftarrow\mathsf{sfEnc}(\mathsf{MPK},1^{|\hat{x_1}|})$ and sends CT to $\mathcal{A}$.

Query phase: $\mathcal{A}$ chooses a circuit $C\in\mathcal{C}_\lambda$ where $C(x_0)=C(x_1)$ and sends C to $\mathcal{I}$. $\mathcal{I}$ computes SK_C as following steps:

1. computes $(p_1,...,p_N)\leftarrow\mathsf{CEncd}(1^\lambda,C)$.
2. constructs an arithmetic circuits C_{p_i} which is equivalent to the polynomial p_i for all $1\le i\le N$.
3. sets $\theta_j=p_j(\hat{x_1})$ for all $1\le j\le N$, where $\hat{x_1}$ is the one computed in **Challenge** phase.
4. sets $\mathsf{sfSK}_{C_{p_j}}\leftarrow\mathsf{sfKG}(\mathsf{MSK},C_{p_j},\theta_j)$ for all $1\le j\le N$.
5. sets $\mathsf{SK}_C=(\mathsf{sfSK}_{C_{p_1}},\dots,\mathsf{sfSK}_{C_{p_N}})$.
6. outputs SK_C.

The security property of RP implies that the difference between Hyb_3 and Hyb_4 is negligible.

Lemma 5. *Assume that* RP *is secure. Then ,*

$$|\mathsf{Pr}[\mathsf{Succ}_4]-\mathsf{Pr}[\mathsf{Succ}_3]|\le2\varepsilon^{\mathsf{RP}}(\lambda).\quad(5)$$

Proof. We construct an adversary $\mathcal{B}_{34}$ against the security of RP. Let $\mathcal{A}$ be an adversary on the games Hyb_3 and Hyb_4. $\mathcal{B}_{34}$ plays a role of the adversary in the game $\mathsf{Exp}^{\mathsf{RP}}_{\mathcal{B}_{34}}(1^\lambda,b)$, and a role of the challenger in the games Hyb_3 and Hyb_4. The description of $\mathcal{B}_{34}$ is as follows.

Setup: $\mathcal{B}_{34}$ generates $(\mathsf{MPK},\mathsf{MSK})\leftarrow\mathsf{FE}_{\mathsf{ours}}.\mathsf{Setup}(1^\lambda)$, then sends MPK to $\mathcal{A}$ as the input.

Challenge: For messages (x_0,x_1) from $\mathcal{A}$, $\mathcal{B}_{34}$ computes $\hat{x_1}\leftarrow\mathsf{InpEncd}(x_1,R)$ for a randomness $R\leftarrow\{0,1\}^{l_R}$. Then $\mathcal{B}_{34}$ computes $\mathsf{CT}=\mathsf{sfCT}\leftarrow\mathsf{sfEnc}(\mathsf{MPK},1^{|\hat{x_1}|})$ and sends CT to $\mathcal{A}$.

Query: For a circuit C from $\mathcal{A}$, $\mathcal{B}_{34}$ computes SK_C as following steps:

1. computes $(p_1, ..., p_N) \leftarrow \mathsf{CEncd}(1^\lambda, C)$.
2. constructs an arithmetic circuits C_{p_j} which is equivalent to the polynomial p_j for all $1 \le j \le N$.
3. sends $(C, x_1, \hat{x_1})$ to the challenger $\mathcal{I}$ of $\mathsf{Exp}^{\mathsf{RP}}_{\mathcal{B}_{34}}(1^\lambda, b)$.
4. receives $(\theta^b_1, \ldots, \theta^b_N)$ from $\mathcal{I}$.
5. computes $\mathsf{sfSK}_{C_{p_j}} \leftarrow \mathsf{sfKG}(\mathsf{MSK}, C_{p_j}, \theta^b_j)$ for all $1 \le j \le N$.
6. sends $\mathsf{SK}_C = (\mathsf{sfSK}_{C_{p_1}}, \ldots, \mathsf{sfSK}_{C_{p_N}})$ to $\mathcal{A}$.

Output: For the output b' of $\mathcal{A}$, $\mathcal{B}_{34}$ outputs b'.

The computation of CT by $\mathcal{B}_{34}$ coincides the one in Hyb_4. Moreover it also coincides the in Hyb_3. This is because $|\hat{x_0}| = |\hat{x_1}|$ follows from $|x_0| = |x_1|$.

For $\mathsf{SK}_C = (\mathsf{sfSK}_{C_{p_1}}, \ldots, \mathsf{sfSK}_{C_{p_N}})$ computed by $\mathcal{B}_{34}$, it follows from Definition 6 that $\mathsf{sfSK}_{C_{p_j}} \leftarrow \mathsf{sfKG}(\mathsf{MSK}, C_{p_j}, \theta^0_j)$ for $\theta^0_j = p_j(\hat{x_1})$ for all $1 \le j \le N$. Namely the computation of SK_C by $\mathcal{B}_{34}$ coincides the one in Hyb_4 if $b = 0$. Therefore, we have

$$\Pr[\mathsf{Exp}^{\mathsf{RP}}_{\mathcal{B}_{34}}(1^\lambda, 0) = 1 \mid \mathcal{B}_{34} \text{ uses } x_1] = \Pr[\mathsf{Succ}_4].$$

Let us consider the case $\mathcal{B}_{34}$ uses x_0 instead of x_1 in **Query** phase. In this case, we have $\mathsf{sfSK}_{C_{p_j}} \leftarrow \mathsf{sfKG}(\mathsf{MSK}, C_{p_j}, \theta^0_j)$ for $\theta^0_j = p_j(\hat{x_0})$. Then the computation of SK_C by $\mathcal{B}_{34}$ when $b = 0$ and x_0 is used coincides the one in Hyb_3. Therefore, we have

$$\Pr[\mathsf{Exp}^{\mathsf{RP}}_{\mathcal{B}_{34}}(1^\lambda, 0) = 1 \mid \mathcal{B}_{34} \text{ uses } x_0] = \Pr[\mathsf{Succ}_3].$$

We consider the case where $b = 1$. When $b = 1$, SK_C computed by $\mathsf{sfSK}_{C_{p_j}} \leftarrow \mathsf{sfKG}(\mathsf{MSK}, C_{p_j}, \theta^1_j)$ for $\theta^1_j = \mathsf{Sim}(1^\lambda, C, C(x_1))$ for all $1 \le j \le N$. Since $C(x_0) = C(x_1)$, the computation of θ^1_j does not changed if x_0 is used instead of x_1. The computation of CT is also unchanged when x_0 is used as we have observed above. Thus we have

$$\Pr[\mathsf{Exp}^{\mathsf{RP}}_{\mathcal{B}_{34}}(1^\lambda, 1) = 1 \mid \mathcal{B}_{34} \text{ uses } x_1] = \Pr[\mathsf{Exp}^{\mathsf{RP}}_{\mathcal{B}_{34}}(1^\lambda, 1) = 1 \mid \mathcal{B}_{34} \text{ uses } x_0].$$

We consider the advantage $\mathsf{Adv}^{\mathsf{RP}}_{\mathcal{B}_{34}}$ when x_1 or x_0 is used. For both cases, we have

$$\begin{aligned}
&\mathsf{Adv}^{\mathsf{RP}}_{\mathcal{B}_{34}}[\mathcal{B}_{34} \text{ uses } x_1] \\
&= |\Pr[\mathsf{Exp}^{\mathsf{RP}}_{\mathcal{B}_{34}}(1^\lambda, 0) = 1 \mid \mathcal{B}_{34} \text{ uses } x_1] - \Pr[\mathsf{Exp}^{\mathsf{RP}}_{\mathcal{B}_{34}}(1^\lambda, 1) = 1 \mid \mathcal{B}_{34} \text{ uses } x_1]| \\
&= |\Pr[\mathsf{Succ}_4] - \Pr[\mathsf{Exp}^{\mathsf{RP}}_{\mathcal{B}_{34}}(1^\lambda, 1) = 1 \mid \mathcal{B}_{34} \text{ uses } x_1]|, \\
&\mathsf{Adv}^{\mathsf{RP}}_{\mathcal{B}_{34}}[\mathcal{B}_{34} \text{ uses } x_0] \\
&= |\Pr[\mathsf{Exp}^{\mathsf{RP}}_{\mathcal{B}_{34}}(1^\lambda, 0) = 1 \mid \mathcal{B}_{34} \text{ uses } x_0] - \Pr[\mathsf{Exp}^{\mathsf{RP}}_{\mathcal{B}_{34}}(1^\lambda, 1) = 1 \mid \mathcal{B}_{34} \text{ uses } x_0]| \\
&= |\Pr[\mathsf{Succ}_3] - \Pr[\mathsf{Exp}^{\mathsf{RP}}_{\mathcal{B}_{34}}(1^\lambda, 1) = 1 \mid \mathcal{B}_{34} \text{ uses } x_0]|.
\end{aligned}$$

By the assumption on the statement that RP is secure, both advantages are bounded the negligible function $\varepsilon^{\mathsf{RP}}$. Thus we have

$$\begin{aligned}
&|\Pr[\mathsf{Succ}_4] - \Pr[\mathsf{Succ}_3]| \\
&\le \mathsf{Adv}^{\mathsf{RP}}_{\mathcal{B}_{34}}[\mathcal{B}_{34} \text{ uses } x_1] + \mathsf{Adv}^{\mathsf{RP}}_{\mathcal{B}_{34}}[\mathcal{B}_{34} \text{ uses } x_0] \\
&\le \varepsilon^{\mathsf{RP}}(\lambda) + \varepsilon^{\mathsf{RP}}(\lambda) = 2\varepsilon^{\mathsf{RP}}(\lambda).
\end{aligned}$$

□

$\mathsf{Hyb}_{5,i}$ $(1 \le i \le N+1)$: $\mathsf{Hyb}_{5,i}$ proceeds in the same way as Hyb_4 except **Challenge** phase and **Query** phase. The description of **Challenge** and **Query** phases in $\mathsf{Hyb}_{5,i}$ is as follows.

Challenge phase: $\mathcal{A}$ chooses two messages $(x_0, x_1) \in \mathcal{X}_\lambda$ and sends them to $\mathcal{I}$. $\mathcal{I}$ computes $\mathsf{CT} \leftarrow \mathsf{FE_{ours}.Enc}(\mathsf{MPK}, x_1)$ and sends CT to $\mathcal{A}$.

Query phase: $\mathcal{A}$ chooses a circuit $C \in \mathcal{C}_\lambda$ where $C(x_0) = C(x_1)$ and sends C to $\mathcal{I}$. $\mathcal{I}$ computes SK_C as following steps:

1. computes $(p_1, ..., p_N) \leftarrow \mathsf{CEncd}(1^\lambda, C)$.
2. constructs an arithmetic circuits C_{p_i} which is equivalent to the polynomial p_i for all $1 \le i \le N$.
3. sets $\theta_j = p_j(\hat{x_1})$ for all $1 \le j \le N$, where $\hat{x_1}$ is the one computed during $\mathsf{FE_{ours}.Enc}(\mathsf{MPK}, x_1)$ in **Challenge** phase.
4. sets $\begin{cases} \mathsf{SK}_{C_{p_j}} \leftarrow \mathsf{PAFE.KeyGen}(\mathsf{MSK}, C_{p_j}) & \text{for } j < i \\ \mathsf{sfSK}_{C_{p_j}} \leftarrow \mathsf{sfKG}(\mathsf{MSK}, C_{p_j}, \theta_j) & \text{for } j \ge i \end{cases}$
5. sets $\mathsf{SK}_C = (\mathsf{SK}_{C_{p_1}}, \ldots, \mathsf{SK}_{C_{p_{i-1}}}, \mathsf{sfSK}_{C_{p_i}}, \ldots, \mathsf{sfSK}_{C_{p_N}})$.
6. outputs SK_C.

The indistinguishability of semi-functional ciphertexts on pkPAFE implies that the difference between Hyb_4 and $\mathsf{Hyb}_{5,1}$ is negligible.

Lemma 6. *Assume that* pkPAFE *has the indistinguishability of semi-functional ciphertexts. Then*

$$|\Pr[\mathsf{Succ}_{5,1}] - \Pr[\mathsf{Succ}_4]| \le \varepsilon^{\mathsf{CTInd}}(\lambda). \tag{6}$$

Proof. We construct an adversary $\mathcal{B}_{45}$ against the indistinguishability of semi-functional ciphertexts of pkPAFE. Let $\mathcal{A}$ be an adversary on the games Hyb_4 and $\mathsf{Hyb}_{5,1}$. $\mathcal{B}_{45}$ plays a role of the adversary in the game $\mathsf{Exp}^{\mathsf{CTInd}}_{\mathcal{B}_{45}}(1^\lambda, b)$, and a role of the challenger in the games Hyb_4 and $\mathsf{Hyb}_{5,1}$. The description of $\mathcal{B}_{45}$ is as follows.

Setup: $\mathcal{B}_{45}$ takes an input MPK from the challenger $\mathcal{I}$ of $\mathsf{Exp}^{\mathsf{CTInd}}_{\mathcal{B}_{45}}(1^\lambda, b)$ then sends MPK to $\mathcal{A}$ as the input.

Challenge: For messages (x_0, x_1) from $\mathcal{A}$, $\mathcal{B}_{45}$ computes $\hat{x_1} \leftarrow \mathsf{InpEncd}(x_1, R)$ for a randomness $R \leftarrow \{0,1\}^{l_R}$. $\mathcal{B}_{45}$ sends $\hat{x_1}$ to the challenger $\mathcal{I}$. After $\mathcal{B}_{45}$ receives CT from $\mathcal{I}$, $\mathcal{B}_{45}$ returns CT to $\mathcal{A}$.

Query: For a circuit C from $\mathcal{A}$, $\mathcal{B}_{45}$ computes SK_C as following steps:

1. computes $(p_1, ..., p_N) \leftarrow \mathsf{CEncd}(1^\lambda, C)$.
2. constructs an arithmetic circuits C_{p_j} which is equivalent to the polynomial p_j for all $1 \le j \le N$.
3. sets $\theta_j = p_j(\hat{x_1})$ for all $1 \le j \le N$, where $\hat{x_1}$ is the one computed in **Challenge** phase.
4. sends $(C_{p_1}, \theta_1, \ldots, C_{p_N}, \theta_N)$ to $\mathcal{I}$.
5. receives SK_C from $\mathcal{I}$, then sends SK_C to $\mathcal{A}$.

Output: For the output b' of $\mathcal{A}$, $\mathcal{B}_{45}$ outputs b'.

For SK_C computed by $\mathcal{B}_{45}$, it follows from Definition 12 that $\mathsf{SK}_C = (\mathsf{sfSK}_{C_{p_1}}, \ldots, \mathsf{sfSK}_{C_{p_N}})$ for $\mathsf{sfSK}_{C_{p_j}} \leftarrow \mathsf{sfKG}(\mathsf{MSK}, C_{p_j}, \theta_j)$ $(1 \le j \le N)$. This coincides with the computation of SK_C in Hyb_4 and $\mathsf{Hyb}_{5,1}$.

For CT, CT is computed by $\mathsf{CT} = \mathsf{sfCT} \leftarrow \mathsf{sfEnc}(\mathsf{MPK}, 1^{|\hat{x_1}|})$ if $b = 0$, or $\mathsf{CT} \leftarrow \mathsf{pkPAFE.Enc}(\mathsf{MPK}, \hat{x_1})$ if $b = 1$ from Definition 12. Therefore $\mathcal{B}_{45}$ coincides the challenger of Hyb_4 when $b = 0$, and coincides the challenger of $\mathsf{Hyb}_{5,1}$ when $b = 1$.

Then we have

$$\begin{aligned}&\mathsf{Pr}[\mathsf{Exp}^{\mathsf{CTInd}}_{\mathcal{B}_{45}}(1^\lambda,0)=1]\\&=\mathsf{Pr}\left[b'=1 \mid \mathsf{CT}=\mathsf{sfCT}\leftarrow\mathsf{sfEnc}(\mathsf{MPK},1^{|\hat{x_1}|})\right]\\&=\mathsf{Pr}[\mathsf{Succ}_4],\end{aligned}$$

and

$$\begin{aligned}&\mathsf{Pr}[\mathsf{Exp}^{\mathsf{CTInd}}_{\mathcal{B}_{45}}(1^\lambda,1)=1]\\&=\mathsf{Pr}\left[b'=1 \mid \mathsf{CT}\leftarrow\mathsf{pkPAFE.Enc}(\mathsf{MPK},\hat{x_1})\right]\\&=\mathsf{Pr}[\mathsf{Succ}_{5,1}].\end{aligned}$$

The advantage $\mathsf{Adv}^{\mathsf{CTInd}}_{\mathcal{B}_{45}}(\lambda)$ of $\mathcal{B}_{45}$ on the indistinguishability of semi-functional ciphertexts is computed by

$$\begin{aligned}\mathsf{Adv}^{\mathsf{CTInd}}_{\mathcal{B}_{45}}(\lambda)&=|\mathsf{Pr}[\mathsf{Exp}^{\mathsf{CTInd}}_{\mathcal{B}_{45}}(1^\lambda,0)=1]-\mathsf{Pr}[\mathsf{Exp}^{\mathsf{CTInd}}_{\mathcal{B}_{45}}(1^\lambda,1)=1]|\\&=|\mathsf{Pr}[\mathsf{Succ}_4]-\mathsf{Pr}[\mathsf{Succ}_{5,1}]|.\end{aligned}$$

By the assumption on the statement that pkPAFE has the indistinguishability of semi-functional ciphertexts, $\mathsf{Adv}^{\mathsf{CTInd}}_{\mathcal{B}_{45}}(\lambda)\le\varepsilon^{\mathsf{CTInd}}(\lambda)$. Thus we have

$$|\mathsf{Pr}[\mathsf{Succ}_4]-\mathsf{Pr}[\mathsf{Succ}_{5,1}]|\le\varepsilon^{\mathsf{CTInd}}(\lambda).$$

□

The indistinguishability of semi-functional keys on pkPAFE implies that the difference among $\mathsf{Hyb}_{5,i}$ is negligible.

Lemma 7. *Assume that* pkPAFE *has the indistinguishability of semi-functional keys. Then for all* $1\le i\le N$*,*

$$|\mathsf{Pr}[\mathsf{Succ}_{5,i}]-\mathsf{Pr}[\mathsf{Succ}_{5,i+1}]|\le\varepsilon^{\mathsf{KeyInd}}(\lambda). \quad (7)$$

Proof. Fix an index i $(1\le i\le N)$. We construct an adversary $\mathcal{B}_5$ against the indistinguishability of semi-functional keys of pkPAFE. Let $\mathcal{A}$ be an adversary on the games $\mathsf{Hyb}_{5,i}$ and $\mathsf{Hyb}_{5,i+1}$. $\mathcal{B}_5$ plays a role of the adversary in the game $\mathsf{Exp}^{\mathsf{KeyInd}}_{\mathcal{B}_5}(1^\lambda,b)$, and a role of the challenger in the games $\mathsf{Hyb}_{5,i}$ and $\mathsf{Hyb}_{5,i+1}$. The description of $\mathcal{B}_5$ is as follows.

Setup: $\mathcal{B}_5$ takes an input MPK from the challenger $\mathcal{I}$ of $\mathsf{Exp}^{\mathsf{KeyInd}}_{\mathcal{B}_5}(1^\lambda,b)$ then sends MPK to $\mathcal{A}$ as the input.

Challenge: For messages (x_0,x_1) from $\mathcal{A}$, $\mathcal{B}_5$ computes $\mathsf{CT}\leftarrow\mathsf{FE}_{\mathsf{ours}}.\mathsf{Enc}(\mathsf{MPK},x_1)$ and sends CT to $\mathcal{A}$.

Query: For a circuit C from $\mathcal{A}$, $\mathcal{B}_5$ computes SK_C as following steps:

1. computes $(p_1,...,p_N)\leftarrow\mathsf{CEncd}(1^\lambda,C)$.
2. constructs an arithmetic circuits C_{p_j} which is equivalent to the polynomial p_j for all $1\le j\le N$.
3. sets $\theta_j=p_j(\hat{x_1})$ for all $1\le j\le N$, where $\hat{x_1}$ is the one computed during $\mathsf{FE}_{\mathsf{ours}}.\mathsf{Enc}(\mathsf{MPK},x_1)$ in **Challenge** phase.
4. sends the following items to the challenger $\mathcal{I}$:
   - circuits and corresponding associated values $(C_{p_{i+1}},\theta_{i+1},\dots,C_{p_N},\theta_N)$,
   - circuits $(C_{p_1},\dots,C_{p_{i-1}})$, and

- the challenge circuit and its associated value (C_{p_i}, θ_i).

5. receives SK_C from $\mathcal{I}$, then sends SK_C to $\mathcal{A}$.

Output: For the output b' of $\mathcal{A}$, $\mathcal{B}_5$ outputs b'.

For SK_C computed by $\mathcal{B}_5$, it follows from Definition 11 that

- $\mathsf{SK}_{C_{p_k}} = \mathsf{sfSK}_{C_{p_k}} \leftarrow \mathsf{sfKG}(\mathsf{MSK}, C_{p_k}, \theta_k)$ for $i+1 \leq k \leq N$,
- $\mathsf{SK}_{C_{p_k}} \leftarrow \mathsf{pkPAFE.KeyGen}(\mathsf{MSK}, C_{p_k})$ for $1 \leq k \leq i-1$,
- $\begin{cases} \mathsf{SK}_{C_{p_i}} = \mathsf{sfSK}_{C_{p_i}} \leftarrow \mathsf{sfKG}(\mathsf{MSK}, C_{p_i}, \theta_i) & \text{if } b=0 \\ \mathsf{SK}_{C_{p_i}} \leftarrow \mathsf{pkPAFE.KeyGen}(\mathsf{MSK}, C_{p_i}) & \text{if } b=1 \end{cases}.$

Namely, $\mathsf{SK}_C = (\mathsf{SK}_{C_{p_1}}, \ldots, \mathsf{SK}_{C_{p_{i-1}}}, \mathsf{sfSK}_{C_{p_i}}, \ldots, \mathsf{sfSK}_{C_{p_N}})$ if $b=0$, or $\mathsf{SK}_C = (\mathsf{SK}_{C_{p_1}}, \ldots, \mathsf{SK}_{C_{p_i}}, \mathsf{sfSK}_{C_{p_{i+1}}}, \ldots, \mathsf{sfSK}_{C_{p_N}})$ otherwise. Therefore $\mathcal{B}_5$ coincides with the challenger of $\mathsf{Hyb}_{5,i}$ when $b=0$, and coincides with the challenger of $\mathsf{Hyb}_{5,i+1}$ when $b=1$.

Then we have

$$\begin{aligned}
&\mathsf{Pr}[\mathsf{Exp}^{\mathsf{KeyInd}}_{\mathcal{B}_5}(1^\lambda, 0) = 1] \\
&= \mathsf{Pr}\left[b' = 1 \mid \mathsf{SK}_C = (\mathsf{SK}_{C_{p_1}}, \ldots, \mathsf{SK}_{C_{p_{i-1}}}, \mathsf{sfSK}_{C_{p_i}}, \ldots, \mathsf{sfSK}_{C_{p_N}})\right] \\
&= \mathsf{Pr}[\mathsf{Succ}_{5,i}],
\end{aligned}$$

and

$$\begin{aligned}
&\mathsf{Pr}[\mathsf{Exp}^{\mathsf{KeyInd}}_{\mathcal{B}_5}(1^\lambda, 1) = 1] \\
&= \mathsf{Pr}\left[b' = 1 \mid \mathsf{SK}_C = (\mathsf{SK}_{C_{p_1}}, \ldots, \mathsf{SK}_{C_{p_i}}, \mathsf{sfSK}_{C_{p_{i+1}}}, \ldots, \mathsf{sfSK}_{C_{p_N}})\right] \\
&= \mathsf{Pr}[\mathsf{Succ}_{5,i+1}].
\end{aligned}$$

The advantage $\mathsf{Adv}^{\mathsf{KeyInd}}_{\mathcal{B}_5}(\lambda)$ of $\mathcal{B}_5$ on the indistinguishability of semi-functional keys is computed by

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{KeyInd}}_{\mathcal{B}_5}(\lambda) &= |\mathsf{Pr}[\mathsf{Exp}^{\mathsf{KeyInd}}_{\mathcal{B}_5}(1^\lambda, 0) = 1] - \mathsf{Pr}[\mathsf{Exp}^{\mathsf{KeyInd}}_{\mathcal{B}_5}(1^\lambda, 1) = 1]| \\
&= |\mathsf{Pr}[\mathsf{Succ}_{5,i}] - \mathsf{Pr}[\mathsf{Succ}_{5,i+1}]|.
\end{aligned}$$

By the assumption on the statement that pkPAFE has the indistinguishability of semi-functional keys, $\mathsf{Adv}^{\mathsf{KeyInd}}_{\mathcal{B}_5}(\lambda) \leq \varepsilon^{\mathsf{KeyInd}}(\lambda)$ follows. Thus we have

$$|\mathsf{Pr}[\mathsf{Succ}_{5,i}] - \mathsf{Pr}[\mathsf{Succ}_{5,i+1}]| \leq \varepsilon^{\mathsf{KeyInd}}(\lambda).$$

□

Hyb_6: Hyb_6 proceeds in the same way as $\mathsf{Hyb}_{5,N+1}$ except **Query** phase. **Query** phase in Hyb_6 is changed as follows.

Query phase: $\mathcal{A}$ chooses a circuit $C \in \mathcal{C}_\lambda$ where $C(x_0) = C(x_1)$ and sends C to $\mathcal{I}$. $\mathcal{I}$ computes $\mathsf{SK}_C \leftarrow \mathsf{FE}_{\mathsf{ours}}.\mathsf{KeyGen}(\mathsf{MSK}, C)$ and sends SK_C to $\mathcal{A}$.

For the difference between $\mathsf{Hyb}_{5,N+1}$ and Hyb_6, we have the following lemma.

Lemma 8.

$$\mathsf{Pr}[\mathsf{Succ}_6] = \mathsf{Pr}[\mathsf{Succ}_{5,N+1}]. \tag{8}$$

Proof. The difference between $\mathsf{Hyb}_{5,N+1}$ and Hyb_6 is **Query** phase. In $\mathsf{Hyb}_{5,N+1}$, all $\mathsf{SK}_{C_{p_j}}$ $(1 \le j \le N)$ is computed as $\mathsf{SK}_{C_{p_j}} \leftarrow \mathsf{PAFE.KeyGen}(\mathsf{MSK}, C_{p_j})$. Then, $\mathsf{SK}_C = (\mathsf{SK}_{C_{p_1}}, \ldots, \mathsf{SK}_{C_{p_N}})$ follows. This means that $\mathsf{Hyb}_{5,N+1}$ coincides with Hyb_6 because $\mathsf{SK}_C \leftarrow \mathsf{FE}_{\mathsf{ours}}.\mathsf{KeyGen}(\mathsf{MSK}, C)$ in Hyb_6. Therefore $\Pr[\mathsf{Succ}_{5,N+1}] = \Pr[\mathsf{Succ}_6]$ follows. □

For the game Hyb_6, the following lemma holds.

Lemma 9.

$$\Pr[\mathsf{Succ}_6] = \Pr[\mathsf{Exp}^{\mathsf{sel}}_{1,\mathcal{A}}(1^\lambda, 1) = 1]. \tag{9}$$

Proof. By Definition 2, Definition 3 and the description of Hyb_6, Hyb_6 coincides with $\mathsf{Exp}^{\mathsf{sel}}_{1,\mathcal{A}}(1^\lambda, 1)$. □

Putting together Eqs. (1), (2), (3), (4), (5), (6), (7), (8), (9), we have

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{sel}}_{\mathcal{A}}(\lambda) &= |\Pr[\mathsf{Exp}^{\mathsf{sel}}_{1,\mathcal{A}}(1^\lambda, 0) = 1] - \Pr[\mathsf{Exp}^{\mathsf{sel}}_{1,\mathcal{A}}(1^\lambda, 1) = 1]| \\
&\le N\varepsilon^{\mathsf{KeyInd}}(\lambda) + \varepsilon^{\mathsf{CTInd}}(\lambda) + 2\varepsilon^{\mathsf{RP}}(\lambda) + \varepsilon^{\mathsf{CTInd}}(\lambda) + N\varepsilon^{\mathsf{KeyInd}}(\lambda) \\
&= 2(N\varepsilon^{\mathsf{KeyInd}}(\lambda) + \varepsilon^{\mathsf{CTInd}}(\lambda) + \varepsilon^{\mathsf{RP}}(\lambda)).
\end{aligned}$$

Since $2(N\varepsilon^{\mathsf{KeyInd}}(\lambda) + \varepsilon^{\mathsf{CTInd}}(\lambda) + \varepsilon^{\mathsf{RP}}(\lambda))$ is negligible in λ, the statement holds. □

5 Conclusion

We have proposed the public-key projective arithmetic functional encryption (pkPAFE). We have given the definition and the syntax of pkPAFE, and shown that pkPAFE derives a public-key functional encryption which is single-key selective secure with the help of the randomizing polynomials scheme. Namely our results imply that the indistinguishability obfuscation can be obtained from pkPAFE.

In this paper, we only consider the abstract notion of pkPAFE and a generic construction of a public-key FE from pkPAFE and RP. Thus an instantiation of pkPAFE is not given like in the case of the secret-key PAFE [3], and every parameters of the resulting FE cannot be set unless the concrete construction of pkPAFE is given. It is an important open question to propose a concrete construction of pkPAFE with the appropriate setting of every parameters.

Acknowledgment

We would like to thank anonymous reviewers for their valuable comments and suggestions. This work was supported in part by JSPS KAKENHI Grant Numbers JP18K11288 and JP19K11956.

References

[1] Prabhanjan Ananth, Aayush Jain, and Amit Sahai. Indistinguishability obfuscation without multilinear maps: io from lwe, bilinear maps, and weak pseudorandomness. Cryptology ePrint Archive, Report 2018/615, 2018. `https://eprint.iacr.org/2018/615`.

[2] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In *Advances in Cryptology – CRYPTO 2015*, pages 308–326. Springer Berlin Heidelberg, 2015.

[3] Prabhanjan Ananth and Amit Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In *Advances in Cryptology – EUROCRYPT 2017*, pages 152–181. Springer International Publishing, 2017.

[4] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. *Computational Complexity*, 15:115–162, 01 2006.

[5] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, pages 1–18, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[6] N. Bitansky and V. Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 171–190, 2015.

[7] Nir Bitansky and Omer Paneth. Zaps and non-interactive witness indistinguishability from indistinguishability obfuscation. In *Theory of Cryptography*, pages 401–427. Springer Berlin Heidelberg, 2015.

[8] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography*, pages 253–273. Springer Berlin Heidelberg, 2011.

[9] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 40–49, 2013.

[10] Shafi Goldwasser and Guy N. Rothblum. On best-possible obfuscation. In Salil P. Vadhan, editor, *Theory of Cryptography*, pages 194–213, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[11] Fuyuki Kitagawa, Ryo Nishimaki, and Keisuke Tanaka. Simple and generic constructions of succinct functional encryption. In Michel Abdalla and Ricardo Dahab, editors, *Public-Key Cryptography – PKC 2018*, pages 187–217, Cham, 2018. Springer International Publishing.

[12] Huijia Lin, Rafael Pass, Karn Seth, and Sidharth Telang. Output-compressing randomized encodings and applications. In *Theory of Cryptography*, pages 96–124. Springer Berlin Heidelberg, 2016.

[13] Huijia Lin and Stefano Tessaro. Indistinguishability obfuscation from trilinear maps and blockwise local prgs. In *Advances in Cryptology – CRYPTO 2017*, pages 630–660. Springer International Publishing, 2017.

[14] Amit Sahai and B. Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. *Proceedings of the Annual ACM Symposium on Theory of Computing*, pages 475–484, 05 2014.