Hindawi
Security and Communication Networks
Volume 2020, Article ID 8823788, 14 pages
https://doi.org/10.1155/2020/8823788

Research Article

Practical CCA-Secure Functional Encryptions for Deterministic Functions

Huige Wang,[1,2] **Kefei Chen**,[3,4] **Tianyu Pan**,[2] **and Yunlei Zhao**[2,5,6]

[1]*Department of Computer, Anhui Science and Technology University, Bengbu 233030, China*
[2]*School of Computer Science, Fudan University, Shanghai 200433, China*
[3]*Department of Mathematics, Hangzhou Normal University, Hangzhou 311121, China*
[4]*Westone Cryptologic Research Center, Beijing 100070, China*
[5]*State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710126, China*
[6]*State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China*

Correspondence should be addressed to Huige Wang; wanghuige@fudan.edu.cn

Received 8 May 2020; Revised 5 June 2020; Accepted 20 July 2020; Published 8 September 2020

Academic Editor: Kaitai Liang

Functional encryption (FE) can implement fine-grained control to encrypted plaintext via permitting users to compute only some specified functions on the encrypted plaintext using private keys with respect to those functions. Recently, many FEs were put forward; nonetheless, most of them cannot resist chosen-ciphertext attacks (CCAs), especially for those in the secret-key settings. This changed with the work, i.e., a generic transformation of public-key functional encryption (PK-FE) from chosen-plaintext (CPA) to chosen-ciphertext (CCA), where the underlying schemes are required to have some special properties such as restricted delegation or verifiability features. However, examples for such underlying schemes with these features have not been found so far. Later, a CCA-secure functional encryption from projective hash functions was proposed, but their scheme only applies to inner product functions. To construct such a scheme, some nontrivial techniques will be needed. Our key contribution in this work is to propose CCA-secure functional encryptions in the PKE and SK environment, respectively. In the existing generic transformation from (adaptively) simulation-based CPA- (SIM-CPA-) secure ones for deterministic functions to (adaptively) simulation-based CCA- (SIM-CCA-) secure ones for randomized functions, whether the schemes were directly applied to CCA settings for deterministic functions is not implied. We give an affirmative answer and derive a SIM-CCA-secure scheme for deterministic functions by making some modifications on it. Again, based on this derived scheme, we also propose an (adaptively) indistinguishable CCA- (IND-CCA-) secure SK-FE for deterministic functions. The final results show that our scheme can be instantiated under both nonstandard assumptions (e.g., hard problems on multilinear maps and indistinguishability obfuscation (IO)) and under standard assumptions (e.g., DDH, RSA, LWE, and LPN).

1. Introduction

PKE is the most useful asymmetric encryption, which encrypted the plaintext using a public encryption key pk and decrypted the ciphertext using a private key sk only owned by the receiver. In particular, in this cryptosystem, without sk, nobody can get any sensitive information about plaintext. This type of cryptosystem does not work well in some special case where fine-grained access control functionality [1, 2] is needed to act on the decrypted plaintext. FE with this functionality was introduced by Boneh et al. [3]. Informally speaking, FE scheme [4–6] can encrypt a message m into a ciphertext ct using a system public key. However, it can only recover a function value $f(m)$ from the ciphertext ct with a private key associated with the function f.

Security is a most important nature of functional encryption, among which indistinguishable chosen-plaintext attack (IND-CPA) security is the most common. IND-CPA security requires that an efficient adversary cannot distinguish the encryption of two different messages even if it allows the adversary to get some extra information such as allowing them access the private key oracle. Furthermore,

according to the capability of the adversary, security can be classified into the selective mode and adaptive mode. In the former, the challenge message must be chosen before the public parameter and private key is generated, while in the latter, it may be chosen at any time.

Constructing FE for general functionalities with adaptive IND-CCA [7] security has been an open problem. Now, expressive functional encryptions with adaptive security have got rapid development. In 2015, Ananth et al. proposed an adaptively secure FE from the selective one [8]. The same year, Waters also presented an adaptively secure FE, but it relies on the strong IO assumption [9–11].

1.1. CCA Security. Although a wealth of results have been obtained on IND-CPA security, those schemes can only resist passive attacks. In order to resist active attacks, an IND-CCA security notion is required which is the same as IND-CPA but with exception that it additionally allows the adversary to access a decryption oracle. IND-CCA security in PKE settings has got richly studied by many researchers. Also, in the IBE [12] and ABE [13] settings, it also got extensively studied. Although IBEs and ABEs are subordinate to functional encryption, none of them could apply to general functions. Thus, studying FE with CCA security for arbitrary functions is necessary. In particular, although we allow the adversary to decrypt any ciphertext with private keys, it obtains by making key queries, and the security equipped with decryption oracle is still stronger than that without decryption oracle. The reason is, for any decryption query (g, ct), the challenger can use the private key sk_g to decrypt ct, even conditioned on $g(m_0) \neq g(m_1)$, but if without decryption oracle, the private key sk_g conditioned on $g(m_0) \neq g(m_1)$ is not allowed to obtain, and thus, ct cannot be decrypted with this key. Simulation-based CCA (SIM-CCA) is a stronger notion than IND-CCA, meaning that the reduction can be simulated via a probability polynomial time (PPT) simulator who only knows partial information and parameters that the system allows.

The FEs with CCA security are presented in [14, 15]. In [14], Benhamouda et al. proposed a CCA-secure functional encryption (FE) that only solves the CCA security in FE based on the projective hash functions. In [15], Nandi and Pandit also solved this problem but with strong restrictions (e.g., needing restricted delegation or verifiability on the underlying IND-CPA FE). However, up to now, it is not known whether the underlying schemes with these special properties exist. Although the generic transformation of Naor and Yung's [16] from IND-CPA to IND-CCA encryption scheme can be adapted to functional encryption, it is only applied to the nonadaptive case in the public-key setting. Of course, it may achieve unbounded collusions, but which is not the key point we focus on in this paper. In particular, in Naor and Yung's generic transformation, the encryption needs to run the original PK-FE twice.

1.2. Our Contributions. To construct functional encryptions with CCA security, some nontrivial techniques will be needed. We propose two (adaptively) CCA-secure generic construction for FE with CCA security for deterministic generic functions where one is in PKE settings and the other is in private-key settings. These generic constructions first yield practical FE that can be instantiated in the standard assumptions.

We first construct an (adaptively) SIM-CCA-secure PK-FE for deterministic functions by making a small modification on Agrawal and Wu's SIM-CCA-secure PK-FE for randomized functions. Then, using the PK-FE scheme together with a NIZK equipped with the property of extractabilities, we construct an (adaptively) IND-CCA-secure SK-FE scheme. Particularly, in our construction, SIM-CPA-secure PK-FE can be used in a black-box way. Thus, the constructed SK-FE can also be based on the same assumption as the underlying PK-FE scheme.

1.3. An Informal Description of Our Schemes. Our constructions are inspired by Agrawal and Wu's generic transformation of PK-FE from deterministic functions to randomized functions [17]. Our solutions can work on standard assumptions and are more efficient than that by Naor and Yung's generic transformation (since we only use the original primitive once rather than twice which is the case in Naor and Yung's generic transformation). They are essentially optimal in communication size: in the derived PK-FE with SIM-CCA security, the public parameter and private key are almost equal to that in the underlying PK-FE, while the ciphertext consists of a PK-FE ciphertext and a NIZK proof. In the SK-FE with IND-CCA security, in addition to a PK-FE ciphertext and a NIZK proof, the final ciphertext also adds an SE ciphertext. In Agrawal and Wu's generic CPA-to-CCA transformation from deterministic functions to randomized functions, they use a simulation-sound extractable NIZK to implement CPA-to-CCA transformation and a Φ-RKA-secure PRF and a one-way permutation to implement deterministic-to-randomized function transformation. Concretely, in the decryption query, e.g., (g, ct), where g is a function, the simulator deals with the decryption by first employing the NIZK to extract a message m from the ciphertext ct and then evaluating function g on m. Thus, the simulator can perfectly realize the decryption functionalities under the extractability of the NIZK. For the deterministic-to-randomized function transformation, the core idea is to use a PRF to remove randomization on the function. Specifically, the scheme transforms the encryption of m to that of the pair $(m, h(k'))$ with randomness $\mathrm{hc}(k')$ under the SIM-CPA FE scheme where h is a one-way permutation and hc is a hardcore function of h. In the construction of the private key for a randomized function f, the scheme first constructs a "derandomize" function $g_k^f(m, h(k')) = f(m, PRF(k \diamond h(k'), m))$ and then computes a SIM-CPA FE private key sk for function g_k^f. The decryption algorithm uses sk to recover $f(m, PRF(k \diamond h(k'), m))$ from a ciphertext ct which encrypts the message $(m, h(k'))$. For our goals of deterministic functions, we remove the PRF and one-way permutation h used for implementing the deterministic-to-randomization function transformation and directly define the final

private key as that of the SIM-CPA PK-FE for deterministic functions. Luckily, the removal operation does not affect the CPA-to-CCA transformation since the *PRF* and h does not participate in the transformation. In this way, we get a SIM-CCA-secure PK-FE scheme for deterministic functions.

In the construction of adaptive IND-CCA SK-FE, a similar methodology proposed by Elkind and Sahai in [18] is adopted. In their construction, the encryption scheme is designed by first constructing a "bare" oblivious-decryptor scheme (ODS) and then employing a designated-verifier NIZK to achieve adaptive CCA security. More specifically, their scheme first encrypted message m under the ODS encryption into a ciphertext c and then employed a designated-verifier NIZK to prove π of $(\mathrm{pk}, c) \in L$.

Compared with their scheme, our work on IND-CCA security for SK-FE uses some seemingly different techniques. In our scheme, beyond a standard simulation-sound NIZK and a PK-FE scheme secure against SIM-CCA, we also use a semantically secure symmetric encryption (SE). The symmetric encryption is used to mask the message directly. Particularly, in order to enable decrypting, the SE ciphertext together with the symmetric key are reencrypted in a PK-FE ciphertext. A simulation-sound NIZK proof functions as a "well-formedness" condition on the symmetric ciphertext and is contained in the overall ciphertext. The underlying PK-FE scheme corresponds to the oblivious-decryptor scheme by Elkind and Sahai [18], whose SIM-CCA security together with the standard NIZK proof is used for rejecting ill-formed ciphertexts.

Note that simulation security is strong; in particular, it implies that we have to restrict ourselves to either secure against bounded collusion, which may hinder the practicability of the scheme, or to simple functionality (for example, IBE), for which the scheme cannot be simulation secure against bounded collusion. Although a weak notion, for example, an IND-CPA-secure SK-FE which may induce a more practical scheme or simple functionality, it is not applicable to our construction. This is because, when we directly use the weak scheme, during the decryption query phase, the master secret key encrypted in the underlying SK-FE ciphertext can be extracted out by the NIZK.

2. Preliminaries

2.1. Noninteractive Zero Knowledge (NIZK). Here, we first explain several notions. $\mathscr{R}: \mathscr{L} \times \mathscr{W} \longrightarrow \{0,1\}$ a PPT binary relation which takes as input a statement $x \in \mathscr{L}$ and the corresponding witness $w \in \mathscr{W}$ and outputs a bit indicating whether x belongs to the language $\mathscr{L}$.

Definition 1 A NIZK system with respect to a relation $\mathscr{R}$ consists of three PPT algorithms $ZK = (ZK.Setup, ZK.Prov, ZK.Ver)$.

Setup: $\mathrm{crs} \longleftarrow ZK.Setup\ (1^\lambda)$, where crs denotes the common random string, which is public and can be accessed by anyone.

Prove: $\pi \longleftarrow ZK.Prov(\mathrm{crs}, x, w)$, where π denotes the proof which proves that it owns the secret witness w of the statement x.

Verification: $b \longleftarrow ZK.Ver(\mathrm{crs}, x, \pi)$, where $b = 1$ denotes the verifier "accept" and $b = 1$ denotes "reject".

Perfect completeness: an argument system ZK is perfectly complete if for all PPT adversaries $\mathscr{A}$, the following holds:

$$Pr\left[\begin{array}{ll} \mathrm{crs} \longleftarrow ZK.Setup\left(1^\lambda\right), & \\ (x,w) \longleftarrow \mathscr{A}(\mathrm{crs}), & : ZK.Ver(\mathrm{crs}, x, \pi) = 1 \\ \pi \longleftarrow ZK.Prov(\mathrm{crs}, x, w) & (x,w) \in \mathscr{R} = 1 \end{array}\right] = 1. \tag{1}$$

Computational soundness: the computational soundness of the scheme ZK is demonstrated as follows, where $\mathscr{A}$ is a PPT algorithm:

$$Pr\left[\mathrm{crs} \longleftarrow ZK.Setup\left(1^\lambda\right), (x,\pi) \longleftarrow \mathscr{A}(\mathrm{crs}): ZK.Ver(\mathrm{crs}, x, \pi) = 1, \text{if} x \notin \mathscr{L}\right] = \mathrm{negl}(\lambda). \tag{2}$$

Definition 2 (zero knowledge). A system $ZK = (ZK.Setup, ZK.Prov, ZK.Ver)$ is said to be computational zero-knowledge if $\exists$ is a PPT simulator $\mathscr{S} = (\mathscr{S}_1, \mathscr{S}_2)$ s.t. for all PPT $\mathscr{A}$, the following holds:

$$\left|Pr\left[\begin{array}{c} \mathrm{crs} \longleftarrow ZK.Setup\left(1^\lambda\right): \\ \mathscr{A}^{\mathscr{O}(\mathrm{crs},\cdot,\cdot)}(\mathrm{crs}) = 1 \end{array}\right]\right| - \left|Pr\left[\begin{array}{c} (\mathrm{crs}, \tau) \longleftarrow \mathscr{S}_1\left(1^\lambda\right): \\ \mathscr{A}^{\mathscr{O}'}(\mathrm{crs}, \tau, \cdot, \cdot)(\mathrm{crs}) = 1 \end{array}\right]\right| = \mathrm{negl}(\lambda), \tag{3}$$

where the oracles $\mathscr{O}(\mathrm{crs}, \cdot, \cdot)$ and $\mathscr{O}'(\mathrm{crs}, \tau, \cdot, \cdot)$ are demonstrated as follows:

(i) The oracle $\mathscr{O}(\mathrm{crs}, \cdot, \cdot)$ takes as input a pair (x, w) and outputs $ZK.Prov(\mathrm{crs}, x, w)$ if $(x, w) \in \mathscr{R}$; o.w. outputs $\perp$.

(ii) The oracle $\mathcal{O}'(\mathrm{crs}, \tau, \cdot, \cdot)$ is the simulator algorithm, which takes as input (x, w) and outputs $\mathcal{S}_2(\mathrm{crs}, \tau, x)$ if $(x, w) \in \mathcal{R}$; o.w. outputs $\perp$.

Definition 3 (simulation-sound extractability). Let $ZK = (ZK.Setup, ZK.Prov, ZK.Ver)$ be a NIZK with respect to a relation $\mathcal{R}$ and $S = (\mathcal{S}_1, \mathcal{S}_2)$ be a PPT simulator. Then, ZK satisfies the notion of simulation-sound extractability if $\exists$ is an extraction algorithm $\xi = (\xi_1, \xi_2)$ s.t.

$$Pr\left[\begin{array}{ll} (\mathrm{crs}, \tau, \varsigma) \longleftarrow \xi_1\left(1^\lambda\right), & (x, \pi) \notin \mathcal{Q} \\ (x, \pi) \longleftarrow \mathcal{A}^{\mathcal{S}_2(\mathrm{crs}, \tau, \cdot)}(\mathrm{crs}), & : \\ w \longleftarrow \xi_2(\mathrm{crs}, \varsigma, x, \pi) & (x, w) \notin \mathcal{R} \\ & ZK.Ver(\mathrm{crs}, x, \pi) = 1 \end{array}\right] = \mathrm{negl}(\lambda), \tag{4}$$

where $\mathcal{Q}$ is the set that contains all queries $\mathcal{A}$ makes to $\mathcal{S}_2$ and the answers from $\mathcal{S}_2$.

2.2. Symmetric Encryption and Semantic Security. Let $SE = (SE.KG, SE.Enc, SE.Dec)$ be a symmetric encryption (SE), $\mathcal{K}_{SE}$ be the secret key space, and $\mathcal{M}_{SE}$ be the message space. These PPT algorithms work as follows: (1) the key-generation algorithm works as $\mathrm{sek} \longleftarrow SE.KG(1^\lambda)$; (2) the encryption algorithm works as $c_{SE} \longleftarrow SE.Enc(\mathrm{sek}, m)$; and (3) the decryption algorithm works as $m \cup \perp \longleftarrow SE.Dec(\mathrm{sek}, c_{SE})$.

Correctness of the scheme is obvious.

Definition 4 (semantic security). An SE is called semantic security (SS), if for all PPT $\mathcal{A}$ in the game $SS_{\mathcal{A}}^{SE,b}(\lambda)$ (see Figure 1) played by $\mathcal{A}$, the function $Adv_{SE,\mathcal{A}}^{SS}$ is negligible in λ (security parameter), where $Adv_{SE,\mathcal{A}}^{SS}(\lambda) = Pr[SS_{\mathcal{A}}^{SE,0}(\lambda) = 1] - Pr[SS_{\mathcal{A}}^{SE,1}(\lambda) = 1] \leq \mathrm{negl}(\lambda)$.

3. PK-FE [8]

Let $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$, $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda \in \mathbb{N}}$, and $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$, where each $f \in \mathcal{F}_\lambda$ takes as input a string $m \in \mathcal{M}_\lambda$ and outputs $f(m) \in \mathcal{Y}_\lambda$.

3.1. Definition of PK-FE. A PK-FE scheme $FE = (FE.Setup, FE.KG, FE.Enc, FE.Dec)$ consists of the following four PPT algorithms:

Setup: the setup algorithm works as $(\mathrm{mpk}, \mathrm{msk}) \longleftarrow FE.Setup(1^\lambda)$, where mpk denotes the master public key and msk denotes the master secret key.

Private key generation: this algorithm works as $\mathrm{sk}_f \longleftarrow FE.KG(\mathrm{msk}, f)$, where $f \in \mathcal{F}_\lambda$ denotes a deterministic function.

Encryption: the encryption algorithm works as $\mathrm{ct} \longleftarrow FE.Enc(\mathrm{mpk}, m)$, where $m \in \mathcal{M}_\lambda$ denotes a plaintext and ct denotes the generated ciphertext.

$\mathrm{SS}_A^{\mathrm{SE},b}(\lambda)$

$\mathrm{sek} \leftarrow \mathrm{SE.KG}(1^\lambda)$

$(m_0, m_1, \mathrm{st}) \leftarrow A(1^\lambda)$

$\mathrm{ct}^* \leftarrow \mathrm{SE.Enc}(\mathrm{sek}, m_b)$

Output $A(\mathrm{ct}^*, \mathrm{st})$

Figure 1: Semantic security for symmetric encryption.

Decryption: the decryption algorithm works as $f(m) \cup \{\perp\} \longleftarrow FE.Dec(\mathrm{sk}_f, \mathrm{ct})$.

Perfect correctness: for all sufficient large $\lambda \in \mathbb{N}$, $(\mathrm{mpk}, \mathrm{msk}) \longleftarrow FE.Setup(1^\lambda)$, $f \in \mathcal{F}_\lambda$, and $m \in \mathcal{M}_\lambda$, the correct implies that

$$Pr[FE.Dec(FE.KG(\mathrm{msk}, f), FE.Enc(\mathrm{mpk}, m)) = f(m)] = 1. \tag{5}$$

3.2. Simulation-Based Chosen-Ciphertext Security. For security, we give the definitions of simulation-based chosen-ciphertext (SIM-CCA) security and indistinguishability-based chosen-ciphertext (IND-CCA) security, respectively. We first give the definition of SIM-CCA security, where $SIM\text{-}CCA_{\mathcal{A},\mathcal{F}}^{FE}(\lambda)$ (see Figure 2) denotes the SIM-CCA game played by a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.

Then, we define the advantage of the adversary $\mathcal{A}$ in game $SIM\text{-}CCA_{\mathcal{A},\mathcal{F}}^{FE}(\lambda)$ as $Adv_{FE,\mathcal{F},\mathcal{A}}^{SIM\text{-}CCA}(\lambda) = Pr[REAL_{\mathcal{A},\mathcal{F}}^{FE}(\lambda) = 1] - Pr[IDEAL_{\mathcal{A},\mathcal{F}}^{FE}(\lambda) = 1]$.

3.2.1. Oracles

(i) *Real Game.* In this game, both oracles $\mathcal{O}_1(\mathrm{msk}, \cdot)$ and $\mathcal{O}_2(\mathrm{msk}, \cdot)$ invoke the algorithm $FE.KG(\mathrm{msk}, \cdot)$ when $\mathcal{A}$ makes key queries to $\mathcal{O}_1(\mathrm{msk}, \cdot)$ and copyright has been updated in VTW, please repush task to validation. Oracle $\mathcal{O}_3(\mathrm{msk}, \cdot, \cdot)$ acts as the decryption oracle, i.e., when $\mathcal{A}$ makes a decryption query $(g, \mathcal{C})$, the oracle $\mathcal{O}_3(\mathrm{msk}, \cdot, \cdot)$ first invokes $FE.KG(\mathrm{msk}, g)$ to obtain sk_g and then computes $y_i = FE.Dec(\mathrm{sk}_g, \mathrm{ct}_i)$ for all $\mathrm{ct}_i \in \mathcal{C}$, where $\mathcal{C} = \{\mathrm{ct}_i\}_{[q_d]}$ and outputs the ordered set $\{y_i\}$. In this game, an ordered set $\{g\}$ is used to denote the functions that the adversary $\mathcal{A}_2$ made queries to $\mathcal{O}_3(\mathrm{msk}, \cdot, \cdot)$ and $\{y\}$ used to denote the answers of the oracle $\mathcal{O}_3$.

(ii) *Ideal Game.* In the ideal game, oracles $\mathcal{O}_1'(\mathrm{st}', \cdot)$ and $\mathcal{O}_2'(\mathrm{st}', \cdot)$ denote the simulator algorithms $\mathcal{S}_2(\mathrm{st}', \cdot)$ and $\mathcal{S}_4(\mathrm{st}', \cdot)$. Simulator $\mathcal{S}_2(\mathrm{st}', \cdot)$ takes a key query f' as input and outputs a key $\mathrm{sk}_{f'}$ for function f'. For simulator $\mathcal{S}_4(\mathrm{st}', \cdot)$, besides given f' as input, it is also allowed to access an oracle $KIdeal(m, \cdot)$, which inputs a function f' and returns $y_i' = f'(m_i)$ for all $i \in [q_c]$. The (ordered) set $\{f'\}$ denotes the functions that the adversary had made key queries to oracles $\mathcal{O}_1'(\mathrm{st}', \cdot)$ and $\mathcal{O}_2'(\mathrm{st}', \cdot)$. Taking $(g', \mathcal{C}')$ as input, oracle $\mathcal{O}_3'(\mathrm{msk}, \cdot, \cdot)$ invokes the simulator $\mathcal{S}_5(\mathrm{st}', \mathrm{ct}_k')$ to obtain m_k' or $\{\perp\}$ for each $\mathrm{ct}_k' \in \mathcal{C}'$.

$\underline{\text{REAL}^{\text{FE}}_{A,F}(\lambda)}$

$(\text{mpk}, \text{msk}) \leftarrow \text{FE.Setup}(1^\lambda)$

$(\text{m}, \text{st}) \leftarrow A_1^{O_1(\text{msk},\cdot),O_3(\text{msk},\cdot,\cdot)}(\text{mpk})$ where $m \in M_\lambda^{\text{q}_c}$

$ct_i^* \leftarrow \text{FE.Enc}(\text{mpk}, m_i)$ for $i \in [q_c]$

$\alpha \leftarrow A_2^{O_2(\text{msk},\cdot),O_4(\text{msk},\cdot)}(\{\text{ct}_i^*\}, \text{st})$

Return $(\text{m}, \{f\}, \{g\}, \{y\}, \alpha)$

$\underline{\text{IDEAL}^{\text{FE}}_{A,F}(\lambda)}$

$(\text{mpk}, \text{st}') \leftarrow S_1(1^\lambda)$

$(\text{m}, \text{st}) \leftarrow A_1^{O_1'(\text{st}',\cdot),O_3'(\text{st}',\cdot,\cdot)}(\text{mpk})$ where $m \in M_\lambda^{\text{q}_c}$

Let $f_1, \cdots, f_{\text{q}1}$ be $A_1's$ queries to $O_1'(\text{st}', \cdot)$

Let $y_{ij} = f_j(\text{m}_i)$ for $i \in [\text{q}_c], j \in [\text{q}_1]$

$(\{\text{ct}_i^*\}, \text{st}') \leftarrow S_3(\text{st}', \{y_{ij}\})$

$\alpha \leftarrow A_2^{O_2'(\text{st}',\cdot),O_3'(\text{st}',\cdot,\cdot)}(\{\text{ct}_i^*\}, \text{st})$

Return $(\text{m}, \{f'\}, \{g'\}, \{y'\}, \alpha)$

Figure 2: SIM − CCA security game for PK − FE.

Next, for each $k \in [q_d]$, if $m_k' = \perp$, then set $y_k' = \perp$; otherwise, compute $y_k' = g'(m_k')$. Finally, it outputs the ordered set $\{y_k'\}$. For convenience, we abbreviate $\{y_k'\}$ as $\{y'\}$.

Here, we give a further restriction on decryption queries. First, we give an equivalence relation notation $\sim$ functioning on ciphertext space $\mathscr{C}$. We say that any two ciphertexts satisfy$\text{ct}_1 \sim \text{ct}_2$, if for all $\lambda \in \mathbb{N}$, $(\text{mpk}, \text{msk}) \longleftarrow FE.Setup(1^\lambda)$ and $\text{sk}_f \longleftarrow FE.KG(\text{msk}, f)$ so that one of the following conditions holds:

(i) $FE.Dec(\text{sk}_f, \text{ct}_1) = \perp$ or $FE.Dec(\text{sk}_f, \text{ct}_2) = \perp$.

(ii) $FE.Dec(\text{sk}_f, \text{ct}_1) = FE.Dec(\text{sk}_f, \text{ct}_2)$.

Next, we give the restriction on decryption queries, namely, for each decryption query $(g, \mathscr{C})$, whether in ideal game or real game, we require that, for all $i, j \in [q_d]$ and $i \neq j$, then $\text{ct}_i \not\sim \text{ct}_j$, and for all $i \in [q_d]$, $j \in [q_c]$, and the challenge ciphertext ct_j^*, it must hold $\text{ct}_i \neq \text{ct}_j^*$.

Definition 5 (SIM-CCA). We say that a PK-FE scheme for deterministic functions is (q_1, q_c, q_2)-simulation-based chosen-ciphertext secure ((q_1, q_c, q_2)-SIM-CCA secure) if for any efficient adversary $\mathscr{A} = (\mathscr{A}_1, \mathscr{A}_2)$ where $\mathscr{A}_1$ and $\mathscr{A}_2$ make at most q_1 and q_2 key-generation queries, respectively, there exists an $\sim$ over ciphertext space ($\sim$) and a PPT simulator $\mathscr{S} = (\mathscr{S}_1, \mathscr{S}_2, \mathscr{S}_3, \mathscr{S}_4, \mathscr{S}_5)$ s.t. the advantage $Adv^{SIM\text{-}CCA}_{FE,\mathscr{F},\mathscr{A}}(\lambda)$ is negligible.

One thing in particular to note is that, in the SIM-CCA security game, if we disallow the adversary $\mathscr{A}$ to query decryption oracles $\mathscr{O}_3$ and $\mathscr{O}_3'$, then the game becomes SIM-CPA game.

3.3. Indistinguishable-Based Chosen-Ciphertext Security. For the IND-CCA security, $IND\text{-}CCA^{FE,b}_{\mathscr{A},\mathscr{F}}(\lambda)$ (see Figure 3) denotes the IND-CCA game played by a PPT adversary $\mathscr{A} = (\mathscr{A}_1, \mathscr{A}_2)$. We define the advantage function $IND\text{-}CCA^{FE,b}_{\mathscr{A},\mathscr{F}}(\lambda)$ as $Adv^{IND\text{-}CCA}_{FE,\mathscr{F},\mathscr{A}}(\lambda) = Pr[IND\text{-}CCA^{FE,0}_{\mathscr{A},\mathscr{F}}(\lambda) = 1] - Pr[IND\text{-}CCA^{FE,1}_{\mathscr{A},\mathscr{F}}(\lambda) = 1]$.

In the above game, we require that if the ordered set $\{f\}$ denotes all key queries and $\{(m_0, m_1)\}$ denotes all challenge message pairs delivered by the adversary $\mathscr{A}$, then the distributions $(\text{mpk}, \text{st}, f(m_0))$ and $(\text{mpk}, \text{st}, f(m_1))$ are identical. Now, we give the definitions of oracles below.

$\underline{\text{IND-CCA}^{\text{FE},b}_{A,F}(\lambda)}$

$(\text{mpk}, \text{msk}) \leftarrow \text{FE.Setup}(1^\lambda)$

$(m_0, m_1, \text{st}) \leftarrow A_1^{\text{KeyGen}(\text{msk},\cdot),\text{O}(\text{msk},\cdot,\cdot)}(\text{mpk})$

where $m_0, m_1 \in M_\lambda$

$\text{ct}^* \leftarrow \text{FE.Enc}(\text{mpk}, m_b)$

Output $A_2^{\text{KeyGen}(\text{msk},\cdot),\text{O}(\text{msk},\cdot)}(\text{ct}^*, \text{st})$

Figure 3: IND − CCA security for PK − FE.

3.3.1. Oracles

(i) $KeyGen(\text{msk}, \cdot)$ denotes a key-generation oracle which, on inputting a function f, invokes the key-generation algorithm $FE.KG(\text{msk}, f)$ to obtain a key sk_f for function f.

(ii) $\mathscr{O}(\text{msk}, \cdot, \cdot)$ denotes a decryption oracle which, on taking as input a tuple (g, ct) such that $\text{ct} \neq \text{ct}^*$, first invokes the key-generation algorithm $FE.KG(\text{msk}, g)$ to obtain a key sk_g and then computes $y = FE.Dec(\text{sk}_g, \text{ct})$ and outputs y.

Note that the above definition is *adaptively* secure, and it is easily modified to the nonadaptive security by requiring the adversary to specify two challenge messages before obtaining the master public key and function keys. In this paper, we only pay attention to the *adaptive* security.

Definition 6 (IND-CCA for PK-FE). We say that a PK-FE scheme for deterministic functions is adaptively $(q_1, 1, q_2)$ indistinguishable chosen-ciphertext secure ($(q_1, 1, q_2)$-IND-CCA secure) if for any PPT adversary $\mathscr{A} = (\mathscr{A}_1, \mathscr{A}_2)$ where $\mathscr{A}_1$ and $\mathscr{A}_2$ make at most q_1 and q_2 key-generation queries, respectively, and we have that the function value $Adv^{IND\text{-}CCA}_{FE,\mathscr{F},\mathscr{A}}(\lambda)$ is negligible.

In particular, as remarked in [19], the above definition can be trivially extended to multiple challenge message security, i.e., adaptive (q_1, q_c, q_2)-IND-CCA security where q_1, q_c, and q_2 are polynomials in λ. Particularly, we stress that the SIM-CCA security described in Section 3.2 implies IND-CCA security.

Lemma 1. *Assume that a PK-FE scheme FE is adaptively SIM-CCA secure, then FE is also adaptively IND-CCA secure.*

The proof is described in Appendix A.

4. Private-Key Functional Encryption

Here, we give the definition of SK-FE for deterministic functions. Let $SFE = (SFE.Setup, SFE.KG, SFE.Enc$, and $SFE.Dec)$ be an SK-FE scheme defined over function space $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ and message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$, respectively. The detailed definition is given as follows:

Setup: the setup algorithm computes $(\text{ppr}, \text{msk}) \longleftarrow SFE.Setup(1^\lambda)$, where ppr denotes the system public parameter ppr, msk denotes the system master secret key, and the public parameter ppr is implied in the following algorithms:

Key generation: the key-generation algorithm computes $\text{sk}_f \longleftarrow SFE.KG(\text{msk}, f)$, where sk_f denotes the generated private key.

Encryption: the encryption algorithm computes $\text{ct} \longleftarrow SFE.Enc(\text{msk}, m)$, where $m \in \mathcal{M}_\lambda$ denotes the being encrypted message and ct denotes the generated ciphertext.

Decryption: the decryption algorithm computes $f(m) \cup \{\perp\} \longleftarrow SFE.Dec(\text{sk}_f, \text{ct})$, where $f(m) \cup \{\perp\}$ denotes the final decrypted result.

Here, we give the perfect correctness of scheme SK-FE, which says that, for all sufficient large $\lambda \in \mathbb{N}$, $(\text{ppr}, \text{msk}) \longleftarrow SFE.Setup(1^\lambda)$, $f \in \mathcal{F}_\lambda$, and $m \in \mathcal{M}_\lambda$, it holds that $Pr[SFE.Dec(SFE.KG(\text{msk}, f), SFE.Enc(\text{msk}, m)) = f(m)] = 1$.

4.1. Indistinguishable-Based Chosen-Ciphertext. Here, we review the IND-CCA of the SK-FE from [8]. In Figure 4, $IND\text{-}CCA^{SFE,b}_{\mathcal{A},\mathcal{F}}(\lambda)$ denotes the IND-CCA game played by a PPT adversary $\mathcal{A}$. The advantage function is defined as $Adv^{IND\text{-}CCA}_{SFE,\mathcal{F},\mathcal{A}}(\lambda) = Pr[IND\text{-}CCA^{SFE,0}_{\mathcal{A},\mathcal{F}}(\lambda) = 1] - Pr[IND\text{-}CCA^{SFE,1}_{\mathcal{A},\mathcal{F}}(\lambda) = 1]$.

In the above game, we require that all challenge messages that the adversary delivers in the encryption queries to oracle ENC satisfy $f(m_0) = f(m_1)$ for all the functions $f \in \mathcal{F}$ that the adversary makes queries to the key-generation oracle $KeyGen$, where m_0 and m_1 denote the challenge message that the adversary delivers in the challenge phase. In the following, we define the functionalities of the oracles.

4.1.1. Oracles

(i) $KeyGen(\text{msk}, \cdot)$ denotes the key-generation oracle which means that when the adversary makes a function query f, the oracle invokes the algorithm $SFE.KG(\text{msk}, f)$ to get a private key sk_f.

(ii) $ENC(\text{msk}, \cdot, \cdot)$ denotes an encryption oracle. On taking as input two messages (m_0, m_1), it invokes $SFE.Enc(\text{msk}, m_b)$ to obtain ct^* for a random bit $b \longleftarrow_\$ \{0, 1\}$. We denote all challenge ciphertexts as $\{\text{ct}^*\}$.

(iii) $\mathcal{O}(\text{msk}, \cdot, \cdot)$ denotes a decryption oracle which, on taking as input a tuple (g, ct) such that $\text{ct} \notin \{\text{ct}^*\}$, invokes the algorithm $SFE.KG(\text{msk}, g)$ to obtain a private key sk_g and outputs $y = SFE.Dec(\text{sk}_g, \text{ct})$.

$\underline{\text{IND-CCA}^{\text{SFE},b}_{A,F}(\lambda)}$
$(\text{ppr}, \text{msk}) \leftarrow \text{SFE.Setup}(1^\lambda)$
Output $A^{\text{KeyGen}(\text{msk},\cdot),\text{ENC}(\text{msk},\cdot,\cdot),\text{O}(\text{msk},\cdot,\cdot)}(1^\lambda, \text{ppr})$

Figure 4: IND − CCA of secret − key functional encryption.

Note that the above definition is *adaptively* secure and is easily modified to the nonadaptively security by requiring the adversary to make encryption queries before key queries.

Definition 7 (IND-CCA for SK-FE). We say that an SK-FE scheme for deterministic functions is adaptively indistinguishability-based chosen-ciphertext secure (adaptively IND-CCA secure) if for all PPT $\mathcal{A}$, the advantage $Adv^{IND\text{-}CCA}_{SFE,\mathcal{F},\mathcal{A}}(\lambda)$ is negligible in λ.

5. Construction of SIM-CCA-Secure PK-FE

In this section, we modify Agrawal and Wu's generic transformation [17] to the setting of deterministic functions.

The modified scheme needs the following building blocks:

(i) A NIZK argument $ZK = (ZK.Setup, ZK.Prov, ZK.Ver)$ with simulation-sound extractability.

(ii) An (adaptively) SIM-CPA-secure PK-FE scheme $CPAFE = (CPAFE.Setup, CPAFE.KG, CPAFE.Enc, CPAFE.Dec)$.

A SIM-CCA-secure PK-FE scheme $CCAFE = (CCAFE.Setup, CCAFE.Enc, CCAFE.KG, CCAFE.Dec)$ is constructed as follows:

(i) The setup algorithm: $(\text{mpk}, \text{msk}) \longleftarrow CCAFE.Setup(1^\lambda)$. This algorithm first runs $(\text{mpk}', \text{msk}') \longleftarrow CPAFE.Setup(1^\lambda)$ and $\text{crs} \longleftarrow ZK.Setup(1^\lambda)$ and then sets $\text{mpk} = (\text{mpk}', \text{crs})$ and $\text{msk} = \text{msk}'$ and releases mpk in public.

(ii) The key-generation algorithm: $\text{sk}_f \longleftarrow CCAFE.KG(\text{mpk}, f)$. This algorithm first parses $\text{mpk} = (\text{mpk}', \text{crs})$ and then runs $\text{sk}'_f \longleftarrow CPAFE.KG(\text{mpk}', f)$ and sets $\text{sk}_f = \text{sk}'_f$.

(iii) The encryption algorithm: on taking as input $\text{mpk} = (\text{mpk}', \text{crs})$ and a message m, this algorithm $CCAFE.Enc(\text{mpk}, m)$ first samples a randomness $r \longleftarrow_\$ \mathcal{R}$ and runs $CPAFE.Enc(\text{mpk}', m; r)$ to get ct' and then computes $\pi \longleftarrow ZK.Prov(\text{crs}, x, w)$, where $x = \text{ct}' \in \mathcal{L}$, the witness $w = (m, r)$, and $(x, w) \in \mathcal{R}$. The relation $\mathcal{R}$ and language $\mathcal{L}$ satisfy the following equation:

$$\exists (m,r) \text{ such that } \mathrm{ct}' = CPAFE.Enc(\mathrm{mpk}', m; r). \quad (6)$$

The final ciphertext is set as $\mathrm{ct} = (\mathrm{ct}', \pi)$.

(iv) The decryption algorithm: on taking as input a private key $\mathrm{sk}_f = \mathrm{sk}_f'$ and a ciphertext $\mathrm{ct} = (\mathrm{ct}', \pi)$, the decryption algorithm first verifies whether $ZK.Ver(\mathrm{crs}, x, \pi) = 1$, where $x = \mathrm{ct}'$; if so, it computes $y = CPAFE.Dec(\mathrm{sk}_f', \mathrm{ct}')$ and outputs y; otherwise, it outputs $\perp$.

It is easy to see that the correctness of the scheme *CCAFE* follows from the completeness of *ZK* and the correctness of *CPAFE*.

Theorem 1. *If ZK is a simulation-sound extractable non-interactive zero-knowledge argument, CPAFE is perfectly correct (q_1, q_c, q_2)-SIM-CPA-secure PK-FE scheme for deterministic function class $\mathcal{F}$, then CCAFE is (q_1, q_c, q_2)-SIM-CCA-secure functional encryption scheme for the same function class.*

Like [17], our definition also requires that the equivalence relation $\sim$ over the ciphertext space $\mathcal{C}$ is satisfied, i.e., given any two ciphertexts $\mathrm{ct}_1 = (\mathrm{ct}_1', \pi_1)$, $\mathrm{ct}_2 = (\mathrm{ct}_2', \pi_2) \in \mathcal{C}$, the equivalence relation $\mathrm{ct}_1 \sim \mathrm{ct}_2$ holds, if $\mathrm{ct}_1' = \mathrm{ct}_2'$.

To prove Theorem 1, we will construct an ideal-world simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4, \mathcal{S}_5)$ for the scheme *CCAFE*. Let $\mathcal{S}^{CPAFE} = (\mathcal{S}_1^{CPAFE}, \mathcal{S}_2^{CPAFE}, \mathcal{S}_3^{CPAFE}, \mathcal{S}_4^{CPAFE})$ be the simulator for the scheme *CPAFE* and $\mathcal{S}^{ZK} = (\mathcal{S}_1^{ZK}, \mathcal{S}_2^{ZK})$ and $\xi^{ZK} = (\xi_1^{ZK}, \xi_2^{ZK})$ be the simulator and extractor for the scheme *ZK*, respectively.

5.1. Simulator $\mathcal{S}_1(1^\lambda)$. The setup simulator $\mathcal{S}_1$ proceeds as follows. On taking as input 1^λ, it first runs $(\mathrm{mpk}', \mathrm{st}_0^{CPAFE}) \longleftarrow \mathcal{S}_1^{CPAFE}$ and then invokes $(\mathrm{crs}, \tau, \varsigma) \longleftarrow \xi_1^{ZK}$, where crs denotes a common random string, τ denotes a simulation trapdoor, and ς denotes an extraction trapdoor. It finally sets $\mathrm{mpk} = (\mathrm{mpk}', \mathrm{crs})$ and $\mathrm{st}_0 = (\mathrm{st}_0^{CPAFE}, \mathrm{mpk}, \tau, \varsigma)$ and outputs $(\mathrm{mpk}, \mathrm{st}_0)$.

5.2. Simulator $\mathcal{S}_2(st_0, f)$. The prechallenge key-generation query simulator $\mathcal{S}_2$ proceeds as follows. On taking as input $\mathrm{st}_0 = (\mathrm{st}_0^{CPAFE}, \mathrm{mpk}, \tau, \varsigma)$ and $f \in \mathcal{F}$, it runs $(\mathrm{sk}_f', \mathrm{st}_1^{CPAFE}) \longleftarrow \mathcal{S}_2^{CPAFE}(\mathrm{st}_0^{CPAFE}, f)$. Then, it outputs the key $\mathrm{sk}_f = \mathrm{sk}_f'$ and a fresh state $\mathrm{st}_1 = (\mathrm{st}_1^{CPAFE}, \mathrm{mpk}, \tau, \varsigma)$.

5.3. Simulator $\mathcal{S}_3(st_0, \{y_{i,j}\}_{i\in[q_c], j\in[q_1]})$. $\mathcal{S}_3$ simulates the challenge ciphertext query procedure. On taking as input a statement $\mathrm{st}_0 = (\mathrm{st}_0^{CPAFE}, \mathrm{mpk}, \tau, \varsigma)$ and a function value set $\{y_{i,j}\}_{i\in[q_c], j\in[q_1]}$, $\mathcal{S}_3$ proceeds as follows. It first computes $(\{\mathrm{c}\ \mathrm{t}_i^*\}_{i\in[q_c]}, \mathrm{s}\ \mathrm{t}_1^{CPAFE}) \longleftarrow \mathcal{S}_3^{CPAFE}(\mathrm{s}\ \mathrm{t}_0^{CPAFE}, \{y_{i,j}\}_{i\in[q_c], j\in[q_1]})$; then for each $i \in [q_c]$, it computes argument $\pi_i^* = \mathcal{S}_2^{ZK}(\mathrm{crs}, \tau, x_i^*)$ with the statement $x_i^* = \mathrm{ct}_i'^*$ described as follows:

$$\exists (m_i, r_i^*) \text{ such that } \mathrm{ct}_i'^* = CPAFE.Enc(\mathrm{mpk}', m_i; r_i^*), \quad (7)$$

where $m = (m_1, \ldots, m_{q_c})$ are the challenge messages that the adversary delivers. At the end of the simulation, it outputs a challenge ciphertext set $\{\mathrm{ct}_i^*\}_{i\in[q_c]}$ and an updated state $\mathrm{st}_1 = (\mathrm{st}_1^{CPAFE}, \mathrm{mpk}, \tau, \varsigma)$, where $\mathrm{ct}_i^* = (\mathrm{ct}_i'^*, \pi_i^*)$.

5.4. Simulator $\mathcal{S}_4(st_0, f)$. $\mathcal{S}_4$ simulates the postchallenge key-generation queries with the help of the oracle $CCAFE.KIdeal(m, \cdot)$. Once getting input $\mathrm{st}_0 = (\mathrm{st}_0^{CPAFE}, \mathrm{mpk}, \tau, \varsigma)$ and $f \in \mathcal{F}$, it first computes $(\mathrm{sk}_f', \mathrm{st}_1^{CPAFE}) \longleftarrow \mathcal{S}_4^{CPAFE}(\mathrm{st}_0^{CPAFE}, f)$ with the help of oracle $CPAFE.KIdeal(m, \cdot)$ and then outputs $\mathrm{sk}_f = \mathrm{sk}_f'$ and $\mathrm{st}_1 = (\mathrm{st}_1^{CPAFE}, \mathrm{mpk}, \tau, \varsigma)$, where the oracle $CPAFE.KIdeal(m, \cdot)$ is defined as follows: for any input function g', it returns $g'(m_i)$ for all $m_i \in m$.

5.5. Simulator $\mathcal{S}_5(st_0, ct)$. The decryption query simulator $\mathcal{S}_5$ takes as input $\mathrm{st}_0 = (\mathrm{st}_0^{CPAFE}, \mathrm{mpk}, \tau, \varsigma)$ and ct, and it parses $\mathrm{mpk} = (\mathrm{mpk}', \mathrm{crs})$ and $\mathrm{ct} = (\mathrm{ct}', \pi)$. Define the statement $x = \mathrm{ct}'$ as

$$\exists (m,r) \text{ such that } \mathrm{ct}' = CPAFE.Enc(\mathrm{mpk}', m; r). \quad (8)$$

If $ZK.Ver(\mathrm{crs}, x, \pi) = 0$, it stops and outputs $\perp$; otherwise, it computes $((m,r) \in \mathcal{M} \times \mathcal{R}) \longleftarrow \xi_2^{ZK}(\mathrm{crs}, \varsigma, x, \pi)$. Finally, it outputs the massage m and state st.

5.6. Hybrid Games. For Theorem 1, we prove it via 4 hybrid games where the first game G_0 is the real-world game and the last game G_3 is the simulator game. Each game contains the following phases:

Setup: in this phase, the challenger generates the system public key mpk'.

Prechallenge queries: in this procedure, $\mathcal{A}$ is allowed to issue the key-generation queries $f \in \mathcal{F}$ and decryption query $(f, \mathrm{ct}) \in \mathcal{F} \times \mathcal{C}$ such that for all decryption queries $(f, \mathcal{C})$ and any ct_i, $\mathrm{ct}_j \in \mathcal{C}$ and $i \neq j$, it holds that $\mathrm{ct}_i \nsim \mathrm{ct}_j$.

Challenge: when the adversary $\mathcal{A}$ submits a vector of messages $m \in \mathcal{M}^{q_c}$ to the challenger, the challenger replies with $\{\mathrm{ct}_i^*\}_{i\in[q_c]}$.

Postchallenge queries: in this procedure, the adversary $\mathcal{A}$ is again allowed to issue key-generation and decryption queries but with a further restriction that decryption query can be only allowed to contain queries $(f, \mathcal{C})$ such that $\mathrm{ct}_j \neq \mathrm{ct}_i^*$, where $\mathrm{ct}_j \in \mathcal{C}$ for $j \in [q_d]$ and ct_i^* is the challenge ciphertext for $i \in [q_c]$.

Output: finally, the adversary $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$ of the challenge bit b.

Each of the hybrid games is described in detail as follows:

Hybrid G_0: this is the game in the real settings where the challenger interacts with the adversary.

Hybrid G_1: same as G_0 except that the way that the common random string crs in the setup phase and the arguments in the challenge ciphertexts are generated. Specifically, in the setup phase, the challenger uses $(\text{crs}, \tau) \longleftarrow \mathcal{S}_1^{ZK}(1^\lambda)$. In the challenge phase, the challenger uses the simulator $\mathcal{S}_2^{ZK}$ to generate the challenge argument in the challenge ciphertexts. Let $m^* \in \mathcal{M}^{q_c}$ be the challenge messages. It samples $r_i^* \in \mathcal{R}$ and computes $\text{ct}_i^{'*} \longleftarrow CPAFE.Enc(\text{mpk}', m_i^*; r_i^*)$ and $\pi_i^* \longleftarrow \mathcal{S}_2^{ZK}(\text{crs}, \tau, x_i^*)$ for $i \in [q_c]$, where the statement x_i^* is described as in equation (1). Finally, the challenger sets $\text{ct}_i^* = (\text{ct}_i^{'*}, \pi_i^*)$ and $\text{mpk} = (\text{mpk}', \text{crs})$ and sends mpk and ct_i^* to the adversary $\mathcal{A}$.

Hybrid G_2: same as G_1 except that the challenger answers the decryption queries by extracting the message-randomness pair (m, r) from the noninteractive zero-knowledge argument ZK. Specifically, in the setup phase, the procedure is same as hybrid G_1 except that the challenger computes $(\text{crs}, \tau, \varsigma) \longleftarrow \xi_1^{ZK}(1^\lambda)$. In the prechallenge phase, the key-generation queries are dealt with as in G_1, while the decryption queries are processed as follows: for query $(g, \mathcal{C})$, where $\mathcal{C} = \{\text{ct}_i\}_{i\in[q_d]}$ and $\text{ct}_i = (\text{ct}_i', \pi_i)$ for $i \in [q_d]$, and let x_i be the statement as described in equation (2). If $ZK.Ver(\text{crs}, x_i, \pi_i) = 0$, set $y_i = \perp$; otherwise, it invokes the extractor $\xi_2^{ZK}(\text{crs}, \varsigma, x_i, \pi_i)$ to obtain a witness (m_i, r_i) and sets $y_i = g(m_i)$. Finally, $\{y_i\}_{i\in[q_d]}$ is sent to $\mathcal{A}$.

Hybrid G_3: same as G_2 but with the difference that the challenger uses the simulator $\mathcal{S}^{CPAFE} = (\mathcal{S}_1^{CPAFE}, \mathcal{S}_2^{CPAFE}, \mathcal{S}_3^{CPAFE}, \mathcal{S}_4^{CPAFE})$ to interact with the adversary. Specifically, in the setup phase, the challenger computes $(\text{mpk}', \text{st}^{CPAFE}) \longleftarrow \mathcal{S}_1^{CPAFE}(1^\lambda)$. In the prechallenge phase, the decryption queries are dealt with as in hybrid G_2, while the key generation is handled as follows: for each query $f \in \mathcal{F}$, it runs $\mathcal{S}_2^{CPAFE}(\text{st}^{CPAFE}, f)$ to obtain private key sk_f. In the challenge phase, let $f_1, \ldots, f_{q_1}$ be the function that the adversary $\mathcal{A}$ has made queries in the key-generation phase and $m \in \mathcal{M}^{q_c}$ be the challenge messages that the adversary queries. It computes $y_{ij} = f_j(m_i)$ for all $i \in [q_c]$ and $j \in [q_1]$ and invokes the simulator $\mathcal{S}_3^{CPAFE}(\text{st}^{CPAFE}, \{y_{i,j}\}_{i\in[q_c], j\in[q_1]})$ to obtain $\{\text{ct}_i^*\}_{i\in[q_c]}$. In the postchallenge queries, the decryption queries are processed as in hybrid G_2, but the key-generation queries are processed differently as below: the challenger invokes $\mathcal{S}_4^{CPAFE}(\text{st}^{CPAFE}, f)$ under the help of the oracle $CPAFE.KIdeal(m, \cdot)$ to obtain the private key sk_f.

Claim 1. If ZK is a computationally noninteractive zero-knowledge argument system, then hybrids G_0 and G_1 are computationally indistinguishable.

Proof. The hybrid games G_0 and G_1 are different in that, in G_1, the challenger uses the NIZK simulator to compute the CRS and arguments. It is easy to see that the claim follows from the computational zero-knowledge of NIZK.

Concretely, let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an efficient adversary distinguishing hybrid games G_0 and G_1. $\mathcal{A}$ is used to construct an algorithm $\mathcal{A}_{ZK}$ that tries to distinguish the real and simulated distributions of the NIZK. $\mathcal{A}_{ZK}$ takes as input crs and is allowed to access to an argument oracle. In the setup, $\mathcal{A}_{ZK}$ computes $(\text{mpk}', \text{msk}') \longleftarrow CPAFE.Setup(1^\lambda)$ and sets $\text{mpk} = (\text{mpk}', \text{crs})$ and $\text{msk} = \text{msk}'$. It then sends mpk to $\mathcal{A}$. In the challenge phase, when $\mathcal{A}$ delivers challenge message $m \in \mathcal{M}_\lambda^{q_c}$, the algorithm $\mathcal{A}_{ZK}$ samples $r_i^* \longleftarrow \mathcal{R}$ and computes $\text{ct}_i^{'*} \longleftarrow CPAFE.Enc(\text{mpk}', m_i; r_i^*)$ for each $i \in [q_c]$. Let x_i^* be the statement in equation (2). $\mathcal{A}_{ZK}$ delivers $(x_i^*, (m_i, r_i^*))$ to its challenger and obtains an argument π_i^*. Finally, $\mathcal{A}$ obtains $ct_i^* = (ct_i^{'*}, \pi_i^*)$ and ct_i^*.

The rest steps are processed in the same way as in G_0 and G_1. Finally, $\mathcal{A}_{ZK}$ outputs whatever $\mathcal{A}$ outputs. Obviously, if the CRS and NIZK arguments are computed honestly, then $\mathcal{A}_{ZK}$ simulates the settings of G_0; otherwise, it simulates the settings of G_1. Thus, the lemma follows. □

Claim 2. If ZK is simulation-sound extractable and $CPAFE$ is perfectly correct, then hybrids G_1 and G_2 are computationally indistinguishable.

Proof. Hybrids G_1 and G_2 are identical but only with the difference in the way the challenger responds to decryption queries. Conditioned on the simulation-sound extractability of ZK and perfect correctness of $CPAFE$, the two games are computationally indistinguishable. Let $(g, \mathcal{C})$ be the decryption query pair that the adversary makes, where $\mathcal{C} = \{\text{ct}_i\}_{i\in[q_d]}$ and $\text{ct}_i = (\text{ct}_i', \pi_i)$.

The procedure can be classified into two cases. If π can pass the verification, the challenger sets $y_i = \perp$ in both G_1 and G_2. Otherwise, it is easy to see that the adversary cannot submit any challenge ciphertexts in the decryption queries. Thus, $\text{ct}_i = (\text{ct}_i', \pi_i)$ was not generated by the challenger using $\mathcal{S}^{ZK}$. Thus, by simulation-sound extractability, with probability at least $1 - \text{negl}(\lambda)$, the extraction algorithm ξ_2^{ZK} on x_i and π_i will produce a witness (m_i, r_i) to make $\text{ct}_i' = CPAFE.Enc(\text{mpk}', m_i; r_i)$ hold. Furthermore, by perfect correctness of $CPAFE$, (m_i, r_i) is the only pair corresponding to ct_i'.

In addition, in G_1, the challenger first computes $\text{sk}_g \longleftarrow CPAFE.KG(\text{msk}', g)$ and then sets $y_i = CPAFE.Dec(\text{sk}_g, \text{ct}_i')$. By the underlying PK-FE scheme $CPAFE$, $y_i = g(m_i)$, which is exactly the output in G_2 after (m_i, r_i) is extracted. Therefore, with probability $1 - \text{negl}(\lambda)$, the distributions in both G_1 and G_2 are identical. □

Claim 3. If $CPAFE$ is a (q_1, q_c, q_2)-SIM-CPA-secure FE scheme for deterministic function family $\mathcal{F}$, then hybrids G_2 and G_3 are computationally indistinguishable.

Proof. Suppose $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is a distinguisher for hybrids G_2 and G_3. $\mathcal{A}$ is used to construct an algorithm $\mathcal{A}^{CPAFE} = (\mathcal{A}_1^{CPAFE}, \mathcal{A}_2^{CPAFE})$ that distinguishes $REAL_{\mathcal{A},\mathcal{F}}^{CPAFE}$ and $IDEAL_{\mathcal{A},\mathcal{F}}^{CPAFE}$. Then, the detailed simulation procedure is processed as follows. □

5.7. The Construction of $\mathcal{A}_1^{CPAFE}$. On taking as input mpk' which is generated by $CPAFE.Setup(1^\lambda)$ or $\mathcal{S}_1^{CPAFE}(1^\lambda)$, the setup and prechallenge queries are simulated as follows:

Setup: in this phase, the adversary $\mathcal{A}_1^{CPAFE}$ first computes a simulated tuple $(\mathrm{crs}, \tau, \varsigma) \longleftarrow xi_1^{ZK}(1^\lambda)$. It then sets $\mathrm{mpk} = (\mathrm{mpk}', \mathrm{crs})$ and sends mpk to $\mathcal{A}$.

Prechallenge queries: in this phase, when $\mathcal{A}_1$ makes a function key query f, the adversary $\mathcal{A}_1^{CPAFE}$ queries its prechallenge key-generation oracle to obtain a key sk_f' which is generated by $\mathrm{sk}_f' = CPAFE.KG(\mathrm{msk}', f)$ or $\mathrm{sk}_f' = \mathcal{S}_2^{CPAFE}(\mathrm{st}', f)$, while the decryption queries are the same as in G_2 and G_3.

5.8. Adversary $\mathcal{A}_2^{CPAFE}$. On taking as input mpk' and a state st from $\mathcal{A}_1^{CPAFE}$, the adversary $\mathcal{A}_2^{CPAFE}$ simulates the challenge phase and postchallenge queries as follows:

Challenge: in this procedure, when the adversary $\mathcal{A}_1$ outputs a challenge message vector m, the adversary $\mathcal{A}_2^{CPAFE}$ queries its challenge oracle to obtain challenge ciphertext set $\{\mathrm{ct}_i'^*\}_{i\in[q_c]}$ for $CPAFE$. It then computes $\pi_i^* \longleftarrow \mathcal{S}_2^{ZK}(\mathrm{crs}, \tau, x_i^*)$ and sets $\mathrm{ct}_i^* = (\mathrm{ct}_i'^*, \pi_i^*)$ for each $i \in [q_c]$, where x_i^* is described as in equation (1). It finally sends $\{\mathrm{ct}_i^*\}_{i\in[q_c]}$ to $\mathcal{A}$.

Prechallenge queries: the simulation of this procedure is the same as prechallenge procedure, except that $\mathcal{A}_2^{CPAFE}$ queries his postchallenge key-generation oracle with the help of oracle $CPAFE.KIdeal(m, \cdot)$ to obtain a key sk_f' which is generated by $\mathrm{sk}_f' = CPAFE.KG(\mathrm{msk}', f)$ or $\mathrm{sk}_f' = \mathcal{S}_4^{CPAFE}(\mathrm{st}', f)$.

At the end, $\mathcal{A}^{CPAFE}$ outputs what $\mathcal{A}$ outputs. Obviously, if $\mathcal{A}^{CPAFE}$ takes part in the game $REAL_{\mathcal{A},\mathcal{F}}^{CPAFE}$, then the view of $\mathcal{A}$ is computationally indistinguishable from G_2; otherwise, if $\mathcal{A}^{CPAFE}$ takes part in the ideal game $IDEAL_{\mathcal{A},\mathcal{F}}^{CPAFE}$, then the view of $\mathcal{A}$ is computationally indistinguishable from G_3. Thus, the proof follows from the (q_1, q_c, q_2)-SIM-CPA security of the scheme $CPAFE$.

Remark 1. Note that applying the above simulation-sound extractable NIZK to the secret-key settings for implementing the IND-CPA-to-IND-CCA security transformation is impossible. This is because beyond the encrypted message, the extractor of the NIZK also extracts out the master secret key used for a secret parameter in the encryption process. Once the extractor obtains the master secret key, it can use it to forge the private keys for any functions of its choice.

In the following, we will use the SIM-CCA-secure PK-FE proposed above and a NIZK scheme to put forward an IND-CCA-secure SK-FE.

6. Construction of SK-FE

Here, we give an IND-CCA-secure SK-FE from SIM-CCA-secure PK-FE. Let $SFE = (SFE.Setup, SFE.KG, SFE.Enc, SFE.Dec)$ be the scheme and $\mathcal{F}$ and $\mathcal{M}_{SFE}$ the associated function space and message space, respectively. The building blocks used in the scheme are as follows:

(i) A noninteractive zero-knowledge argument $ZK = (ZK.Setup, ZK.Prov, ZK.Ver)$ over the common random string space $\{0,1\}^{|\mathrm{crs}|}$ and proof space Π.

(ii) A semantically secure symmetric encryption scheme $SE = (SE.KG, SE.Enc, SE.Dec)$ over the key space $\mathcal{K}_{SE}$, message space $\mathcal{M}_{SE}$, and ciphertext space $\mathcal{C}_{SE}$.

(iii) A SIM-CCA-secure PK-FE scheme $FE = (FE.Setup, FE.KG, FE.Enc, FE.Dec)$ over the message space $\mathcal{M}_{FE} = \{0,1\}^{|\mathrm{crs}|} \times \mathcal{C}_{SE} \times \Pi \times \mathcal{K}_{SE}$ and function space $U[\mathcal{F}]$.

In particular, note that in this construction, the message space of the scheme SFE is identical to that of the scheme SE, namely, $\mathcal{M}_{SFE} = \mathcal{M}_{SE}$.

Setup: taking 1^λ as input, the algorithm $SFE.Setup(1^\lambda)$ first computes $(\mathrm{fpk}, \mathrm{fsk}) \longleftarrow FE.Setup(1^\lambda)$, $\mathrm{sek} \longleftarrow SE.KG(1^\lambda)$, and $\mathrm{crs} \longleftarrow ZK.Setup(1^\lambda)$. Then, it outputs the public parameter $\mathrm{ppr} = (\mathrm{crs}, \mathrm{fpk})$ and master secret key $\mathrm{msk} = (\mathrm{fsk}, \mathrm{sek})$. Note that the parameter ppr is implicitly contained in the following algorithms.

Encryption: taking a master secret key msk and a message $m \in \mathcal{M}_{SE}$ as input, the algorithm $SFE.Enc(\mathrm{msk}, m)$ first parses msk into $\mathrm{msk} = (\mathrm{fsk}, \mathrm{sek})$. It then computes a symmetric encryption ciphertext $\mathrm{ct}_{SE} \longleftarrow SE.Enc(\mathrm{sek}, m)$, an argument $\pi \longleftarrow ZK.Prov(\mathrm{crs}, \mathrm{ct}_{SE}, (\mathrm{sek}, m))$, and a PK-FE ciphertext $\mathrm{ct}_{FE} \longleftarrow FE.Enc(\mathrm{fpk}, \mathrm{ct}_{SE}, \mathrm{sek})$. It finally outputs the overall ciphertext $\mathrm{CT} = (\mathrm{ct}_{SE}, \pi, \mathrm{ct}_{FE})$.

Key generation: taking a master secret key msk and a function $f \in \mathcal{F}$ as input, the algorithm $SFE.KG(\mathrm{msk}, f)$ first parses msk into $\mathrm{msk} = (\mathrm{fsk}, \mathrm{sek})$ and then constructs a universal circuit $U[f]$ (see Figure 5) and computes $\mathrm{sk}_f \longleftarrow FE.KG(\mathrm{fsk}, U[f])$. It finally outputs the private key sk_f.

Decryption: taking a private key $\mathrm{sk}_f = \mathrm{sk}_{U[f]}$ and a ciphertext $\mathrm{CT} = (\mathrm{ct}_{SE}, \pi, \mathrm{ct}_{FE})$ as input, the algorithm $SFE.Dec(\mathrm{sk}_f, \mathrm{CT})$ first verifies whether $ZK.Ver(\mathrm{crs}, \mathrm{ct}_{SE}, \pi) = 1$; if so, it computes and outputs $y = FE.Dec(\mathrm{sk}_{U[f]}, \mathrm{ct}_{FE})$; otherwise, it outputs $\perp$.

The correctness follows from Theorem 2.

Theorem 2. *If FE is perfectly correct, SE is a correct SE scheme, ZK is a perfectly complete NIZK, and the circuit $U[f]$ is correctly defined, then the scheme SFE is a correct SK-FE scheme.*

Proof. It is easily observed the correctness of SFE follows from the perfect-completeness of ZK, the perfect correctness of FE, the correctness of SE, and the correct definition of the circuit $U[f]$. □

Proof. Intuition. Our construction is inspired by Agrawal and Wu's generic transformation for PK-FE from deterministic functions to randomized functions [17]. From their scheme, we first derive an SIM-CCA-secure PK-FE for

Constant : f
Input : $\mathrm{ct}_{\mathrm{SE}}$, sek
(1) Compute $m = \mathrm{SE.Dec}(\mathrm{sek}, \mathrm{ct}_{\mathrm{SE}})$.
(2) Output $f(m)$.

Figure 5: Circuit $U[f]$.

deterministic functions by removing Φ-RKA-secure pseudorandom function PRF and one-way permutation used to implement the deterministic-to-randomized function transformation (note that the details of Agrawal and Wu's scheme can be found in [17], we will not dwell on them here). Then using the derived PK-FE together with a semantically secure symmetric encryption (SE) and a standard NIZK, we obtain an SK-FE scheme with IND-CCA security. Roughly speaking, the plaintext is masked with the SE scheme (where the symmetric key of the SE is a part of the master secret key), then it computes a proof under the NIZK which is used for verifying whether the SE ciphertext encrypts correct plaintext. In order to ensure correct decryption, we reencrypt the SE ciphertext together with the proof and the symmetric key under the derived PK-FE scheme, in which the symmetric key and the SE ciphertext make sure that the scheme enables to decrypt and the proof makes sure that ill-formed SE ciphertext can be rejected in the final scheme.

The security follows Theorem 3. □

Theorem 3. *If the NIZK argument ZK is computationally indistinguishable, the SE scheme SE is semantically secure, the PK-FE scheme FE is perfectly correct (adaptively) (q_1, q_c, q_2)-SIM-CCA secure, then the SK-FE scheme SFE described above is (adaptively) IND-CCA secure, where the adversary makes at most $q_1 + q_2$ key queries and q_c challenge queries.*

Proof. We adopt a similar proof method as that of Theorem 1. Likewise, each game contains the following procedures:

Setup: the challenger computes the parameter ppr.

Key generation query: in this procedure, the adversary $\mathcal{A}$ may make prechallenge key queries and postchallenge key queries for functions $f \in \mathcal{F}$. Particularly, note that, for all key queries $\{f_j\}_{j\in[q_1+q_2]}$ and encryption queries $\{(m_0^i, m_1^i)\}_{i\in[q_c]}$ as below, we require that $f_j(m_0^i) = f(m_1^i)$.

Encryption query: in this procedure, $\mathcal{A}$ commits a collection of message pairs $\{(m_0^i, m_1^i)\}_{i\in[q_c]}$, and the challenger responds with $\{ct_i^*\}_{i\in[q_c]}$.

Decryption query: here, $\mathcal{A}$ may make decryption queries (g, CT), with a further restriction that decryption queries are only allowed to contain queries (g, CT) such that $\mathrm{CT} \notin \mathcal{Q}$, where $\mathcal{Q}$ is the set of all encryption queries that $\mathcal{A}$ has made in the encryption query procedure.

In particular, since in the underlying PK-FE scheme FE, for any function $U[f]$ and message pair $(m_0' = (\mathrm{ct}_{SE}^0, \mathrm{sek}^0), m_1' = (\mathrm{ct}_{SE}^1, \mathrm{sek}^1))$ (assume ct_{SE}^0 is an SE encryption of message m_0 under key sek^0 and ct_{SE}^1 is an SE encryption of message m_1 under key sek^1, where f and (m_0, m_1) are the function and message pair that the IND-CCA adversary of the SK-FE scheme delivers in key query and encryption query phases) that the SIM-CCA adversary delivers in key query and encryption query phases, and it satisfies $U[f](m_0') = U[f](m_1')$, namely, $f(m_0) = f(m_1)$. Thus, the admissible condition still holds in our SK-FE scheme. Furthermore, like in the PK-FE, we also require that a similar equation relation $\sim$ over ciphertext space $\mathcal{C}$ is satisfied, i.e., given any two ciphertexts $\mathrm{CT}_1 = (\mathrm{ct}_{SE}^1, \pi^1, \mathrm{ct}_{FE}^1)$ and $\mathrm{CT}_2 = (\mathrm{ct}_{SE}^2, \pi^2, \mathrm{ct}_{FE}^2)$, we say $\mathrm{CT}_1 \sim \mathrm{CT}_2$, if $\mathrm{ct}_{SE}^1 = \mathrm{ct}_{SE}^2$. □

6.1. Hybrid Games

G_0: this is the premier game where the message m_0 is encrypted in the encryption query phase.

G_1: same as G_0 except that the generation of the common random string crs and the argument π are replaced with the simulation mode of the scheme ZK.

G_2: same as G_1 but with the difference that the simulator $\mathcal{S}^{FE} = (\mathcal{S}_1^{FE}, \mathcal{S}_2^{FE}, \mathcal{S}_3^{FE}, \mathcal{S}_4^{FE}, \mathcal{S}_5^{FE})$ is used to respond to the queries.

G_3: same as G_2 but with the difference that challenge message m_1 instead of m_0 is encrypted in the challenge ciphertext. In particular, the symmetric encryption computes the encryption $\mathrm{ct}_{SE} \longleftarrow SE.Enc(\mathrm{sek}, m_1)$ of m_1.

G_4: this is the same as G_3 except that the challenger uses the real algorithms of the scheme FE to respond to the queries.

G_5: this is the same as G_4 except that the generation of the common random string crs and argument π is replaced with the real algorithms of the scheme ZK.

Claim 4. If the NIZK argument ZK is computational zero knowledge, then hybrids G_0 and G_1 are computationally indistinguishable.

(i) At the setup phase, the adversary $\mathcal{A}_{ZK}$ first samples $(\mathrm{fpk}', \mathrm{fsk}') \longleftarrow FE.Setup(1^\lambda)$ and $\mathrm{sek} \longleftarrow SE.KG(1^\lambda)$. Then, it sets $\mathrm{ppr} = (\mathrm{crs}, \mathrm{fpk}')$ and $\mathrm{msk} = (\mathrm{fsk}', \mathrm{sek})$.

(ii) In the key-generation procedure, when a key query $f \in \mathcal{F}_\lambda$ is queried by $\mathcal{A}$, the adversary $\mathcal{A}_{ZK}$ first constructs a universal circuit $U[f]$ and then responds with $\mathrm{sk}_f \longleftarrow FE.KG(\mathrm{fsk}', U[f])$.

(iii) During the encryption query, when $\mathcal{A}$ makes an encryption query (m_0, m_1), the adversary $\mathcal{A}_{ZK}$ first computes a symmetric encryption $\mathrm{ct}_{SE}^* \longleftarrow SE.Enc(\mathrm{sek}, m_0)$. It then uses $(\mathrm{ct}_{SE}^*, (\mathrm{sek}, m_0))$ to query the oracle $\mathcal{O}(\mathrm{crs}, \cdot, \cdot)$ and obtains an argument π^* (which is generated by the real algorithm $ZK.Prov$ or the simulation algorithm $\mathcal{S}_2^{ZK}$). It next computes a PK-FE ciphertext

$\mathrm{ct}_{FE}^{*} \longleftarrow FE.Enc(\mathrm{fmk}', \mathrm{ct}_{SE}^{*}, \mathrm{sek})$. Finally, it sends $\mathrm{CT}^{*} = (\mathrm{ct}_{SE}^{*}, \pi^{*}, \mathrm{ct}_{FE}^{*})$ to $\mathcal{A}$. Let $\mathcal{Q}$ denote the set of all challenge ciphertexts that $\mathcal{A}$ obtains via encryption queries.

(iv) In the decryption query phase, when $\mathcal{A}$ makes (g, CT) query s.t. $\mathrm{CT} \notin \mathcal{Q}$, the adversary $\mathcal{A}_{ZK}$ first parses CT into $\mathrm{CT} = (\mathrm{ct}_{SE}, \pi, \mathrm{ct}_{FE})$ and verifies whether $ZK.Ver(\mathrm{crs}, \mathrm{ct}_{SE}, \pi) = 1$. If so, it generates sk_g, computes, and outputs $y = FE.Dec(\mathrm{sk}_g, \mathrm{ct}_{FE})$ to $\mathcal{A}$.

Proof. Assume G_0 and G_1 are distinguishable, then we can use the adversary $\mathcal{A}$ that distinguishes G_0 and G_1 to construct an adversary $\mathcal{A}_{ZK}$ that attempts to break the computational zero-knowledge property of ZK. $\mathcal{A}_{ZK}$ is given a common random string crs (which comes from the real algorithm or simulation algorithm) and may access to the oracle $\mathcal{O}(\mathrm{crs}, \cdot, \cdot)$, which computes the argument for statements belonging to the language:

At the end, $\mathcal{A}_{ZK}$ outputs whatever $\mathcal{A}$ outputs.

From above, we can see that the adversary $\mathcal{A}_{ZK}$ perfectly simulates the environment for $\mathcal{A}$. Obviously, if crs and π are from the real algorithm, then $\mathcal{A}_{ZK}$ simulates the hybrid G_0 for $\mathcal{A}$; otherwise, it simulates G_1 for $\mathcal{A}$.

Claim 5. If the PK-FE scheme FE is (adaptively) (q_1, q_c, q_2)-SIM-CCA secure, then hybrids G_1 and G_2 are computationally indistinguishable.

Proof. Assume that G_1 and G_2 are distinguishable, then an adversary $\mathcal{A}$ that distinguishes G_1 and G_2 can be used to construct an adversary $\mathcal{A}^{FE} = (\mathcal{A}_1^{FE}, \mathcal{A}_2^{FE})$ which tries to break through the SIM-CCA security of the scheme FE; that is, the experiments $REAL$ and $IDEAL$ can be distinguished. In the security proof, $\mathcal{A}^{FE}$ will play the role of the challenger for the adversary $\mathcal{A}$. Let $\mathcal{O}_{KG}^{pre}$ be the prechallenge key-generation oracle corresponding to the oracle $\mathcal{O}_1(\mathrm{fsk}', \cdot)$ in the real experiment and the oracle $\mathcal{O}_1'(\mathrm{st}', \cdot)$ in the ideal experiment, $\mathcal{O}_{KG}^{post}$ be the postchallenge oracle corresponding to the oracle $\mathcal{O}_2(\mathrm{fsk}', \cdot)$ in the real experiment and the oracle $\mathcal{O}_2'(\mathrm{st}', \cdot)$ in the ideal experiment, and $\mathcal{O}_{dec}$ be the decryption oracle corresponding to the oracle $\mathcal{O}_3(\mathrm{fsk}', \cdot)$ in the real experiment and the oracle $\mathcal{O}_3'(\mathrm{st}', \cdot)$ in the ideal experiment. The reduction is described as follows:

(i) At the begin, $\mathcal{A}_1^{FE}$ first gets fpk'. It then samples $\mathrm{sek} \longleftarrow SE.KG(1^{\lambda})$ and $(\mathrm{crs}, \tau) \longleftarrow \mathcal{S}_1^{ZK}(1^{\lambda})$ and finally sets the public parameter $\mathrm{ppr} = (\mathrm{crs}, \mathrm{fpk}')$.

(ii) For the key-generation query, when $\mathcal{A}$ makes a prechallenge key query $f \in \mathcal{F}_{\lambda}$, $\mathcal{A}_1^{FE}$ first constructs a universal circuit $U[f]$ and then makes a query $U[f]$ to the oracle $\mathcal{O}_{KG}^{pre}$, from which it obtains a key sk_f' (which is generated by $\mathrm{sk}_f' \longleftarrow FE.KG(\mathrm{fsk}', U[f])$ or $\mathrm{sk}_f' \longleftarrow \mathcal{S}_2^{FE}(\mathrm{st}', U[f])$). When $\mathcal{A}$ makes a postchallenge key query $f \in \mathcal{F}_{\lambda}$, the adversary $\mathcal{A}_2^{FE}$ queries $U[f]$ to the oracle $\mathcal{O}_{KG}^{post}$ to obtain a key $\mathrm{sk}_f = \mathrm{sk}_f'$ and sends it to $\mathcal{A}$ (note that sk_f' is generated by $\mathrm{sk}_f' \longleftarrow FE.KG(\mathrm{fsk}', U[f])$ or $\mathrm{sk}_f' \longleftarrow \mathcal{S}_2^{FE}(\mathrm{st}', U[f])$).

(iii) During the encryption query, when $\mathcal{A}$ makes an encryption query (m_0, m_1), the adversary $\mathcal{A}_2^{FE}$ first computes a symmetric encryption $\mathrm{ct}_{SE}^{*} \longleftarrow SE.Enc(\mathrm{sek}, m_0)$ and an argument $\pi^{*} \longleftarrow \mathcal{S}_2^{ZK}(\mathrm{crs}, \tau, \mathrm{ct}_{SE}^{*})$. It then makes a challenge query $(\mathrm{ct}_{SE}^{*}, \mathrm{sek})$ to its challenger, from which it obtains a PK-FE ciphertext ct_{FE}^{*} (which is generated by $\mathrm{ct}_{FE}^{*} \longleftarrow FE.Enc(\mathrm{fpk}', \mathrm{ct}_{SE}^{*}, \mathrm{sek})$ or $\mathrm{ct}_{FE}^{*} \longleftarrow \mathcal{S}_3^{FE}(\mathrm{st}', \{f_j(m_0)\}_{j \in [q_1]})$, where $\{f_j(m_0)\}_{j \in [q_1]}$ are the outputs of all functions $U[f_j]$ on $(\mathrm{ct}_{SE}^{*}, \mathrm{sek})$, and the set $\{f_j\}_{j \in [q_1]}$ refers to all functions that the adversary $\mathcal{A}_1^{FE}$ makes queries to the oracle $\mathcal{O}_{KG}^{pre}$). Finally, it sends the ciphertext $\mathrm{CT}^{*} = (\mathrm{ct}_{SE}^{*}, \pi^{*}, \mathrm{ct}_{FE}^{*})$ to the adversary $\mathcal{A}$. Let $\mathcal{Q}$ be the set of all responses of this phase. Note that the adversary can make at most q_c encryption queries in this phase.

(iv) In the decryption query phase, when $\mathcal{A}$ makes a query (g, CT) s.t. $\mathrm{CT} \notin \mathcal{Q}$ (for both prechallenge decryption queries and postchallenge decryption queries), the adversary $\mathcal{A}_2^{FE}$ first parses CT into $\mathrm{CT} = (\mathrm{ct}_{SE}, \pi, \mathrm{ct}_{FE})$. It then verifies whether $ZK.Ver(\mathrm{crs}, \mathrm{ct}_{SE}, \pi) = 1$. If the verification passes, a query (g, ct_{FE}) is made to the oracle $\mathcal{O}_{dec}$ (for both prechallenge decryption queries and postchallenge decryption queries) to get the result y and sends it to $\mathcal{A}$ (note that y is generated by the real decryption algorithm or the simulation algorithm of the scheme FE).

At the end, $\mathcal{A}^{FE}$ outputs whatever $\mathcal{A}$ outputs.

From above, we can observe that the environment for $\mathcal{A}$ is perfectly simulated by the adversary $\mathcal{A}^{FE}$. Obviously, if the master public key fpk', the private keys sk_f', the challenge ciphertexts ct_{FE}^{*}, and the value y are generated as in real experiment, then $\mathcal{A}^{FE}$ simulates hybrid G_1 for $\mathcal{A}$; otherwise, it simulates hybrid G_2 for $\mathcal{A}$. Thus, if the adversary $\mathcal{A}$ distinguishes hybrids G_1 and G_2, then the adversary $\mathcal{A}^{FE}$ breaks the (q_1, q_c, q_2)-SIM-CCA security of FE. The claims are as follows. □

Claim 6. If the SE scheme SE is semantically secure, then games G_2 and G_3 are computationally indistinguishable.

Proof. Assume that G_2 and G_3 are distinguishable, then an adversary $\mathcal{A}$ can be used to construct an algorithm $\mathcal{A}_{SE}$ that attempts to break through the semantic security of the scheme SE. In particular, the adversary is allowed to access the oracle $\mathcal{O}(\mathrm{sek}, \cdot)$ that can generate ciphertexts for messages that the adversary chooses. Let $\mathcal{B}$ be the challenger of the adversary $\mathcal{A}_{SE}$.

(i) In the setup phase, the adversary $\mathcal{A}_{SE}$ samples $(\mathrm{fpk}', \mathrm{st}') \longleftarrow S_1^{FE}(1^{\lambda})$ and $(\mathrm{crs}, \tau) \longleftarrow \mathcal{S}_1^{ZK}(1^{\lambda})$ and sets $\mathrm{ppr} = (\mathrm{crs}, \mathrm{fpk}')$.

(ii) In the key-generation phase, the adversary $\mathcal{A}_{SE}$ deals with the function key queries for $\mathcal{A}$ as in hybrid G_3.

(iii) In the encryption phase, when $\mathcal{A}$ queries the challenge plaintext pair (m_0, m_1), the adversary $\mathcal{A}_{SE}$

first delivers a pair (m_0, m_1) to $\mathcal{B}$ (its challenger) to get a symmetric ciphertext ct^*_{SE} (which encrypts the message m_b for a challenge bit $b \leftarrow\!\!\$ \{0,1\}$). It then computes an argument $\pi^* \leftarrow\!\!- \mathcal{S}_2^{ZK}(\mathrm{crs}, \tau, \mathrm{ct}^*_{SE})$ and an FE ciphertext $\mathrm{ct}^*_{FE} \leftarrow\!\!- \mathcal{S}_3^{FE}(\mathrm{st}', \{f_j(m_b)\}_{j\in[q_1]})$ (note that $f_j(m_0) = f_j(m_1)$), where the set $\{f_j\}_{j\in[q_1]}$ refers to the functions that the adversary $\mathcal{A}_1^{FE}$ makes queries to the oracle $\mathcal{O}_{KG}^{pre}$. In the end, it sends the ciphertext $\mathrm{CT}^* = (\mathrm{ct}^*_{SE}, \pi^*, \mathrm{ct}^*_{FE})$ to $\mathcal{A}$.

Finally, $\mathcal{A}^{SE}$ outputs whatever $\mathcal{A}$ outputs.

From above, we can observe that the adversary $\mathcal{A}^{SE}$ perfectly simulates the environment for $\mathcal{A}$. Obviously, if the challenge ciphertext CT^* encrypts the message m_0, then $\mathcal{A}^{SE}$ simulates hybrid G_2 for $\mathcal{A}$; if it encrypts the message m_1, then $\mathcal{A}^{SE}$ simulates hybrid G_3 for $\mathcal{A}$. Thus, if the adversary $\mathcal{A}$ distinguishes hybrids G_2 and G_3, then the adversary $\mathcal{A}^{SE}$ breaks through the security of SE. □

Claim 7. If the PK-FE scheme FE is SIM-CCA secure, then hybrids G_3 and G_4 are computationally indistinguishable.

Proof. The same as that of Claim 5. □

Claim 8. If the NIZK argument ZK is computationally zero-knowledge, then hybrids G_4 and G_5 are computationally indistinguishable.

Proof. The same as that of Claim 4. □

Remark 2. Note that, since the generic function f in the SK-FE scheme is hardwired in the circuit $U[f]$ in the underlying PK-FE, the transformation in this paper is restricted to security against bounded collusion only. Removing this restriction by allowing only very specific functions seems violating our goals for genetic functions. Bounded collusion appears inherent in this type of FE schemes and solving it seems not easy. We left this work as an open problem and hope to solve it in the future.

7. Instantiation

7.1. Instantiations of the Primitives. Both primitives used by our generic construction can be designed under some standard assumptions. For example, the simulation-sound extractable NIZK arguments can be instantiated from RSA assumption, and the SE scheme can be instantiated from one-way functions.

7.1.1. Simulation-Sound Extractable NIZK Arguments. Agrawal and Wu [17] showed that the simulation-sound extractable NIZK argument can be constructed from trapdoor one-way permutations and dense cryptosystems by De Santis et al. [20] and both of which can be instantiated from RSA assumption.

7.1.2. Semantically Secure Symmetric Encryption. We note that the semantically secure symmetric encryption SE can be instantiated from one-way functions [21]. For example, it can be constructed from weak pseudorandom functions (wPRF) PRF, namely, $SE.Enc(\mathrm{sek}, m; r) = (r, PRF(\mathrm{sek}, r) \oplus m)$, and this wPRF can be again designed from only one-way functions.

7.2. Instantiations of the Underlying PK-FE. Here, we show several examples of how to instantiate a perfectly correct (q_1, q_c, q_2)-SIM-CPA-secure functional encryption scheme $CPAFE$ to apply to our generic constructions with CCA security in both public key and private key settings. The results described below show that the SIM-CPA-secure PK-FE used to construct the SIM-CCA-secure PK-FE can be designed from both standard assumptions (such as DDH, RSA, LWE, and LPN) and nonstandard assumptions (such as IO and intractable problems on composite-order multilinear maps).

7.2.1. SIM-CPA-Secure PK-FE from Standard Assumptions. As in [22], Gorbunov et al. give a generic construction of PK-FE for a restricted number of secret key queries. Specifically, they proposed two $(q_1, 1, \mathrm{poly})$ and $(q_1, \mathrm{poly}, 0)$-SIM-CPA-secure PK-FE schemes for any class of deterministic functions computable by polynomial-size circuits based on the CPA-secure PKE and PRG computable by low-degree circuit. Particularly, both the PKE and PRG can be instantiated from standard assumptions. Concretely, the former can be designed based on the standard assumptions such as RSA [23], DDH [24], LWE [25], and LPN [26] assumptions, while the latter can be instantiated from the RSA assumption. Given the instantiated SIM-CPA-secure PK-FE, we obtain a SIM-CCA-secure PK-FE. Then using the SIM-CCA-secure PK-FE, we obtain an IND-CCA-secure SK-FE instantiable from the same assumptions.

7.2.2. SIM-CPA-Secure PK-FE from Nonstandard Assumptions. As in [17], an adaptively IND-CPA-secure PK-FE with overwhelming correctness from intractable assumptions on composite-order multilinear maps by Garg et al. [10] can be applied to construct a PK-FE with SIM-CPA security by employing the overwhelming-to-perfect correctness transformation proposed by Dwork et al. [27] and the IND-to-SIM security transformation proposed by Caro et al. [28], respectively. In addition, SIM-CPA-secure PK-FE can also be obtained by applying the selective-to-adaptive transformation by Ananth et al. [8] and the IND-to-SIM transformation by Caro et al. [28] to the IO-based selectively-IND-secure PK-FE by Garg et al. [29]. Thus, given the instantiated SIM-CPA-secure PK-FE from nonstandard assumptions, we can obtain SIM-CCA-secure PK-FE and IND-CCA-secure SK-FE and both of which can be based on the nonstandard assumptions.

8. Conclusions and Open Problems

Our works propose two adaptively CCA-secure FEs in the PKE and SKE settings, respectively. The concrete works are

as follows. We first derive an/a (adaptively) SIM-CCA-secure FE for deterministic functionalities in the public-key settings by making modifications on ones for randomized functionalities proposed by Agraval and Wu [17]. Then, based on the modified scheme together with a symmetric encryption and a standard NIZK, we present a private-key FE scheme (SK-FE) with adaptive IND-CCA security for deterministic functionalities. The instantiabilities of the underlying base SIM-CPA-secure PK-FEs, symmetric encryption, and NIZK from standard assumptions show that our proposed SK-FE scheme is practical under the standard assumptions.

8.1. IND-CCA-Secure SK-FE for Randomized Functionalities. A precursor to the work on SK-FE scheme supporting randomized functionalities was the work by Komargodski et al. [30]. They achieve this by utilizing function-private FE scheme for deterministic functionalities in the private-key settings. Brakerski and Segev [31] proposed a function-private scheme based on any SK-FE scheme for a sufficiently rich function class. To construct SK-FE with IND-CCA security for randomized functionalities, a $function-private$ FE in the public-key settings with SIM-CCA security for randomized functionalities may be necessary, but deriving such a scheme from existing works is nontrivial, and it may require some new techniques. Thus, we left them as an open problem.

Appendix

A. SIM-CCA Security Implies IND-CCA Security

Proof of Lemma 1. Assume that there exists a PPT IND-CCA adversary $\mathcal{A}^I = (\mathcal{A}_1^I, \mathcal{A}_2^I)$ that can break the IND-CCA security of the PK-FE scheme *FE*; then, there exists a SIM-CCA adversary $\mathcal{A}^S = (\mathcal{A}_1^S, \mathcal{A}_2^S)$ that can break the SIM-CCA security of scheme *FE*. We now use the IND-CCA adversary $\mathcal{A}^I = (\mathcal{A}_1^I, \mathcal{A}_2^I)$ to construct the SIM-CCA adversary $\mathcal{A}^S = (\mathcal{A}_1^S, \mathcal{A}_2^S)$. The adversary $\mathcal{A}^S$ is constructed as follows.

Setup phase: in this phase, the SIM-CCA adversary $\mathcal{A}_1^S$ receives a master public key mpk from its challenger $\mathcal{B}$. It then sends mpk to the IND-CCA adversary $\mathcal{A}_1^I$.

Prechallenge queries: in this phase, when the IND-CCA adversary $\mathcal{A}_1^I$ makes a function key query $f \in \mathcal{F}_\lambda$, the SIM-CCA adversary $\mathcal{A}_1^S$ queries its prechallenge key-generation oracle to obtain a key sk_f which is generated as in the real mode or in the simulation mode. For the decryption query $(g, \mathcal{C})$, $\mathcal{A}_1^S$ queries its prechallenge decryption oracle to obtain $\{y_i\}$ and sends $\{y_i\}$ to the adversary $\mathcal{A}_1^I$.

Challenge queries: in this phase, once the adversary $\mathcal{A}_1^I$ outputs two challenge messages m_0, m_1, and state st (which is sent to the adversary $\mathcal{A}_2^I$), $\mathcal{A}_2^S$ chooses $b \longleftarrow \{0,1\}$ and outputs m_b as the challenge message of the SIM-CCA security. Then, it queries its challenge oracle to obtain the challenge ciphertext set $\{ct^*\}$ which is generated as in the real mode or in the ideal mode. Finally, it sends $\{ct^*\}$ to the adversary $\mathcal{A}_2^I$.

Postchallenge queries: in this phase, the key-generation queries and decryption queries are dealt with in the same way as in the prechallenge phase except that when the postchallenge key-generation oracle of the SIM-CCA adversary $\mathcal{A}_2^S$ works in the ideal mode, it also needs an additional help of oracle *KIdeal*.

At the end of the experiment, the SIM-CCA adversary $\mathcal{A}_2^S$ outputs what the IND-CCA adversary $\mathcal{A}_2^I$ outputs.

It is easy to see that if the adversary $\mathcal{A}^I$ is valid, i.e., for all functions $\{f\}$ and messages m that $\mathcal{A}^I$ delivers in function key queries and message queries, we have $f(m_0) = f(m_1)$. By this, the simulators $\mathcal{S}_3(\mathrm{st}', f(m_0))$ and $\mathcal{S}_3(\mathrm{st}', f(m_1))$ in the SIM-CCA security game are computationally indistinguishable. Thus, $\mathcal{A}^S$ perfectly simulates the attacking environment for $\mathcal{A}^I$. Finally, we conclude that if the adversary $\mathcal{A}^I$ can break the IND-CCA security of the scheme *FE*, then the adversary $\mathcal{A}_2^S$ can break the SIM-CCA security with almost the same probability.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The first author was supported by the National Natural Science Foundation of China (Grant nos. NSFC61702007 and NSFC61572318), National Key Research and Development Program of China (Grant no. 2017YFB0802000), and other foundations (Grant nos. 2019M661360(KLH2301024), gxbjZD27, KJ2018A0533, XWWD201801, ahnis20178002, KJ2017A519, 16ZB0140, LD14127X, and ZRC2013380). The second author was supported by the National Key Research and Development Program of China (Grant no. 2017YFB0802000) and National Natural Science Foundation of China (Grant no. NSFC61472114). The fourth author was supported in part by the National Key Research and Development Program of China (Grant no. 2017YFB0802000), National Natural Science Foundation of China (Grant nos. 61877011, 61472084, and U1536205), Shanghai Innovation Action Project (Grant no. 16DZ1100200), Shanghai Science and Technology Development Funds (Grant no. 16JC1400801), and Shandong Provincial Key Research and Development Program of China (Grant nos. 2017CXG0701 and 2018CXGC0701).

References

[1] H. Wang, K. Chen, J. K. Liu, Z. Hu, and Y. Long, "Access control encryption with efficient verifiable sanitized decryption," *Information Sciences*, vol. 465, pp. 72–85, 2018.

[2] H. G. Wang, K. F. Chen, B. D. Qin, X. J. Lai, and Y. H. Wen, "A new construction on randomized message-locked

encryption in the standard model via uces," *Science China Information Sciences*, vol. 60, no. 5, 2017.

[3] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: definitions and challenges," in *Advance in TCC 2011*, Y. Ishai, Ed., pp. 253–273, Springer, Berlin, Germany, 2011.

[4] H. Wang, K. Chen, J. K. Liu, and Z. Hu, "Leakage-resilient chosen-ciphertext secure functional encryption from garbled circuits," *Information Security Practice and Experience*, pp. 119–140, 2018.

[5] H. G. Wang, K. F. Chen, Y. Zhang, and Y. L. Zhao, "Functional encryption with application to machine learning: simple conversions from generic functions to quadratic functions," *Peer-to-Peer Networking and Applications*, 2020.

[6] H. G. Wang, Y. Zhang, K. Chen, G. Y. Sui, Y. L. Zhao, and X. Y. huang, "Functional broadcast encryption with applications to data sharing for cloud storage," *Information Sciences*, vol. 502, pp. 109–124, 2019.

[7] H. G. Wang, K. F. Chen, B. D. Qin, and Z. Y. Hu, "LR-RRA-CCA secure functional encryption for randomized functionalities from trapdoor HPS and LAF," *Science China Information Sciences*, vol. 61, no. 5, 2018.

[8] P. Ananth, Z. Brakerski, G. Segev, and V. Vaikuntanathan, "From selective to adaptive security in functional encryption," in *Advance in CRYPTO 2015*, R. Gennaro and M. Robshaw, Eds., pp. 657–677, Springer, Berlin, Germany, 2015.

[9] 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 2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, IEEE, Berkeley, CA, USA, pp. 40–49, 2013.

[10] S. Garg, C. Gentry, S. Halevi, and M. Zhandry, "Functional encryption without obfuscation," in *Proceedings of the TCC 2016*, pp. 480–511, Beijing, China, 2016.

[11] B. Waters, "A punctured programming approach to adaptively secure functional encryption," in *Lecture Notes in Computer Science*, pp. 678–697, Springer, Berlin, Germany, 2015.

[12] E. Kiltz and Y. Vahlis, "CCA2 secure IBE: standard model efficiency through authenticated symmetric encryption," in *CT-RSA 2008*Springer, Berlin, Germany, 2008.

[13] S. Yamada, N. Attrapadung, G. Hanaoka, and N. Kunihiro, "Generic constructions for chosen-ciphertext secure attribute based encryption," in *Public Key Cryptography-PKC 2011*, pp. 71–89, Springer, Berlin, Germany, 2011.

[14] F. Benhamouda, F. Bourse, and H. Lipmaa, "CCA-secure inner-product functional encryption from projective hash functions," in *PKC 2017*, pp. 36–66, Springer, Berlin, Germany, 2017.

[15] M. Nandi and T. Pandit, "Generic conversions from CPA to CCA secure functional encryption," International Association for Cryptologic Research, Lyon, France, 2015, Report 2015/457, https://eprint.iacr.org/2015/457.pdf.

[16] M. Naor and M. Yung, "Public-key cryptosystems provably secure against chosen ciphertext attacks," in *Proceedings of the ACM Symposium on the Theory of Computing*, pp. 427–437, Baltimore, MD, USA, 1990.

[17] S. Agrawal and D. J. Wu, "Functional encryption: deterministic to randomized functions from simple assumptions," in *Lecture Notes in Computer Science*Springer, Berlin, Germany, 2016, http://eprint.iacr.org/2016/482.pdf.

[18] E. Elkind and A. Sahai, "A unified methodology for constructing public-key encryption schemes secure against adaptive chosen-ciphertext attack (2002)," in *IACR Cryptology Eprint Archive*University of California, Berkeley, CA, USA, 2002.

[19] V. Goyal, A. Jain, V. Koppula, and A. Sahai, "Functional encryption for randomized functionalities," in *TCC 2015, Part II, LNCS*, Y. Dodis, J.B. Nielsen, Eds., vol. 9015, pp. 325–351, Springer, Berlin, Germany, 2015.

[20] A. D. Santis, G. D. Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai, "Robust non-interactive zero knowledge," in *Proceedings of the CRYPTO 2001*, pp. 566–598, Santa Barbara, CA, USA, 2001.

[21] O. Goldreich, *Foundations of Cryptography—Volume 2: Basic Applications*Cambridge University Press, Cambridge, UK, 2004.

[22] S. Gorbunov, V. Vaikuntanathan, and H. Wee, "Functional encryption with bounded collusions via multi-rty computation," in *Advances in Cryptology-RYPTO 2012*, R. Safavi-aini and R. Canetti, Eds., vol. 7417, pp. 162–179, Springer, Berlin, Germany, 2012.

[23] D. Pointcheval, "RSA public-key encryption," in *Encyclopedia of Cryptography and Security*, Springer, Berlin, Germany, 2011.

[24] S. J. Li, F. T. Zhang, Y. X. Sun, and L. M. Shen, "Efficient leakage-resilient public key encryption from ddh assumption," *Cluster Computing*, vol. 16, no. 4, pp. 797–806, 2013.

[25] X. C. Sun, B. Li, X. H. Lu, and F. Y. Fang, "CCA secure public key encryption scheme based on LWE without Gaussian sampling," in *Proceedings of the International Conference on Information Security and Cryptology*, pp. 361–378, Beijing, China, 2015.

[26] Y. Yu and J. Zhang, "Cryptography with auxiliary input and trapdoor from constant-noise LPN," in *Advances in Cryptology-CRYPTO 2016*, pp. 214–243, Springer, Berlin, Germany, 2016.

[27] C. Dwork, M. Naor, and O. Reingold, "Immunizing encryption schemes from decryption errors," in *EUROCRYPT 2004*, pp. 342–360, Springer, Berlin, Germany, 2004.

[28] A. D. Caro, V. Iovino, A. Jain, A. O Neill, O. Paneth, and G. Persiano, "On the achievability of simulation-based security for functional encryption," in *CRYPTO 2013*, pp. 519–535, Springer, Berlin, Germany, 2013.

[29] S. Garg, G. Craig, S. Halevi, M. Raykova, S. Amit, and B. Waters, "Candidate indistinguishability obfuscation and functional encryption for all circuits," in *Proceedings of the 54th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Berkeley, CA, USA, pp. 40–49, 2013.

[30] I. Komargodski, G. Segev, and E. Yogev, "Functional encryption for randomized functionalities in the privateey setting from minimal assumptions," in *TCC 2015*, Y. Dodis and J.B. Nielsen, Eds., vol. 9015, pp. 352–377, Springer, Berlin, Germany, 2015.

[31] Z. Brakerski and G. Segev, "Function-private functional encryption in the private-key setting," *Journal of Cryptology*, vol. 31, no. 1, pp. 202–225, 2018.