Chosen Ciphertext Secure Functional Encryption from Constrained Witness PRF

Tapas Pal, Ratna Dutta

Department of Mathematics, Indian Institute of Technology Kharagpur
Kharagpur-721302, India
tapas.pal@iitkgp.ac.in,ratna@maths.iitkgp.ac.in

Abstract. Functional encryption generates sophisticated keys for users so that they can learn specific functions of the encrypted message. We provide a generic construction of chosen ciphertext attacks (CCA) secure public-key functional encryption (PKFE) for all polynomial-size circuits. Our PKFE produces succinct ciphertexts that are independent of the size and depth of the circuit class under consideration.

We accomplish our goal in two steps. First, we define a new cryptographic tool called *constrained witness pseudorandom function* (CWPRF) which is motivated by combining WPRF of Zhandry (TCC 2016) and constrained PRF of Boneh and Waters (ASIACRYPT 2013). More specifically, CWPRF computes pseudorandom values associated with NP statements and generates constrained keys for boolean functions. We can recompute the pseudorandom value corresponding to a particular statement either using a public evaluation key with a valid witness for the statement or applying a constrained key for a function that satisfies the statement. We construct CWPRF by coupling indistinguishability obfuscation ($i\mathcal{O}$) and CPRF supporting all polynomial-size functions.

In the second and main technical step, we show a generic construction of a CCA secure PKFE for all circuits utilizing our CWPRF. It has been observed that obtaining PKFE supporting all circuits is already a complex task and $i\mathcal{O}$-based constructions of PKFEs are only proven to be chosen plaintext attacks (CPA) secure. On the other hand, existing CCA secure functional encryption schemes are designed for specific functions such as equality testing, membership testing, linear function etc. We emphasize that our construction presents the first CCA secure PKFE for all circuits along with succinct ciphertexts.

Keywords: constrained witness pseudorandom function, functional encryption, obfuscation.

1 Introduction

An essential research trend in cryptography is to investigate relationships among existing primitives and establish concrete security for the primitives that have several valuable applications. Exploring such correlations provide new insights concerning the structure and security of the considered primitive. Consequently,

generic approaches in building cryptographic primitives is a significant aspect of research.

In this paper, we generically construct a public-key functional encryption (PKFE) scheme for all polynomial-size functions that is secure against active adversaries. The concept of PKFE is being formalized by Boneh, Sahai and Waters [10]. The main importance of functional encryption (FE) lies in the fact that it simply subsumes most of the advanced public-key primitives including identity-based encryption (IBE), attribute-based encryption (ABE) and predicate encryption (PE). The goal of PKFE is to generate secret-keys dedicated to a class of functions so that a particular secret-key enables a user to learn a specific function of the publicly encrypted message, but remains oblivious about the plain message.

Traditionally, encryption schemes are constructed to satisfy indistinguishability against chosen plaintext attacks (IND-CPA) where the adversary is not given access to the decryption oracle. Intuitively, IND-CPA security for PKFE [10] guarantees that an adversary can not distinguish between encryption of messages m_0 and m_1 even when it has polynomially many secret-keys for functions f satisfying $f(m_0) = f(m_1)$. However, over time the cryptographic community has shifted towards achieving indistinguishability against chosen ciphertext attacks (IND-CCA) for many FE schemes [12,32,21] — one of the main reasons is the fact that IND-CCA security withstands against attackers that can make decryption queries to keys it did not ask before and hence the encryption becomes non-malleable [13,1]. We refer [31] for an exceptional discussion on the significance of IND-CCA security.

To fulfil the goal of achieving IND-CCA secure PKFE, we can either generically transform existing IND-CPA secure PKFEs into IND-CCA secure schemes or we directly construct IND-CCA secure PKFE schemes. However, the direct construction of IND-CCA secure FE has been rarely studied in the literature. In case of generic transformation, one of the efficient approaches is the Fujisaki-Okamoto [14] transformation. Although it induces very low ciphertext overhead, the security is proved in the random oracle model [2]. Another option is to follow Naor-Yung dual encryption technique [25] which appends a non-interactive zero-knowledge (NIZK) proof [6,29] determining the ciphertext is well-formed. This approach is expensive as we need to compute NIZK proof of the encryption circuit in a gate-by-gate manner. Moreover, the ciphertext overhead is quite high as the proof size grows linearly with the size of the encryption circuit.

These techniques have been well studied in the context of plain public-key encryption (PKE). While there are a number of research for IND-CCA secure PKE [13,12,23,20], very little can be found on IND-CCA secure FE. The aim of all prior works was centred in demonstrating IND-CCA secure FE for specific function classes, e.g. IBE (equality testing) [19], ABE (membership testing) [21], PE (certain relation circuits) [5,21] and inner product FE (linear functions) [3]. Evidently, IND-CPA secure PKFE for all circuits is already quite complex [15,18] to achieve and new cryptographic tools or techniques are required to realize the stronger security.

Our Results. In this work, we explore a direct generic construction of IND-CCA secure PKFE scheme for all polynomial size boolean functions. To reach our goal, we first formalize a new cryptographic tool called *constrained witness pseudorandom function* (CWPRF). We give construction of CWPRF using an indistinguishability obfuscation ($i\mathcal{O}$) [15] and a constrained pseudorandom function (CPRF) [11].

Formalization of CWPRF. Zhandry introduced the notion of WPRF [34] to generate pseudorandom values associated to statements of an NP language L. More precisely, WPRF has a secret function key fk and a public evaluation key ek such that the secret-key fk is used to compute a pseudorandom value $y = \mathsf{F}(\mathsf{fk}, x)$ for any statement x (of a fixed length) and the public-key ek along with a witness w helps to recover y if the witness proves that x is in the language. A constrained WPRF is a natural extension of normal WPRF. For a circuit C, a CWPRF is capable of generating a constrained key fk_C using the secret-key fk so that fk_C enables one to produce the pseudorandom value $\mathsf{F}(\mathsf{fk}, x)$ if $C(x) = 1$. Thus, CWPRF provides finer access control to the pseudorandom values as we can embed any functionality into the constrained keys.

Security of CWPRF. The security notions of CWPRF are defined to combine the security of two related primitives CPRF [11] and WPRF [34]. Mainly, we consider two flavors of security: pseudorandomness and function privacy. The CWPRF is said to satisfy pseudorandomness at a given statement $x \notin L$ if an adversary is unable to distinguish $\mathsf{F}(\mathsf{fk}, x)$ from a random element even when the adversary gets polynomially many F-values $\mathsf{F}(\mathsf{fk}, x')$ for statements $x' \neq x$ and many constrained keys fk_C such that $C(x) = 0$. The function privacy of CWPRF ensures that an adversary, given oracle access to $\mathsf{F}(\mathsf{fk}, \cdot)$, cannot distinguish between two constrained keys fk_{C_0} and fk_{C_1} for two different circuits C_0 and C_1 unless the keys are trivially separated. A formal discussion on security of CWPRF is given in Sec. 2.3.

Construction of CWPRF. We provide a generic construction of CWPRF using indistinguishability obfuscation ($i\mathcal{O}$) and CPRF. Informally, $i\mathcal{O}$ makes a program unintelligible in a way that the obfuscated program preserves the functionality of the original program. Our CWPRF is built upon the $i\mathcal{O}$-based CPRF of Boneh et al. [9] where they have used subexponential security of $i\mathcal{O}$ to achieve function privacy. However, our application requires only a weak version of function privacy for CWPRF and fortunately, the underlying CPRF of [9] satisfies this weak function privacy assuming a polynomially secure $i\mathcal{O}$ (Remark D.11 of [8]). In weak function privacy, the adversary's ability of distinguishing between two constrained keys fk_{C_0} and fk_{C_1} is negligible whenever C_0 and C_1 are equivalent circuits. Therefore, a polynomially secure $i\mathcal{O}$ [18] is sufficient for our CWPRF (described in Sec. 3) and its applications mentioned below.

CCA secure PKFE. To demonstrate the power of CWPRF, we describe a generic construction of CCA secure PKFE for all polynomial size boolean circuits. The building strategy is inspired by the simulation secure secret-key FE (SKFE) given by Boneh, Kim and Montgomery [7] in the context of proving

that simulation based function privacy is impossible to achieve for CPRF. We emphasize that this impossibility result is restricted to simulation based privacy whereas our work deals with indistinguishability based privacy of the primitives.

We utilize our CWPRF and the puncturable WPRF (PWPRF) proposed by Pal and Dutta [28] to build the PKFE. Note that, a PWPRF is a restricted class of CWPRF where the constrains are point functions. Specifically, the psudorandomness of PWPRF and the weak function privacy of CWPRF are employed to realize our full adaptive CCA secure PKFE in Sec. 4. Apart from CCA security, our PKFE enjoys an optimal size ciphertext which has not been achieved before for FEs that supports all polynomial size circuits. In particular, encryption of a message m has a size of $|m|+ \text{poly}(\lambda)$ where $|m|$ denotes the bit-size of m and λ is the security parameter. The optimality of the PKFE implies succinctness of ciphertexts [15] meaning that the size of ciphertexts is independent of the circuit sizes or even the depths. Existing PKFE for all circuits [15] (based on $i\mathcal{O}$) which satisfies such succinctness property does not achieve optimality of ciphertexts or strong CCA security.

In an additional application, we utilize the pseudorandomness property of CWPRF to develop a tag-based CCA secure ABE for all circuits. Similar to the PKFE, our ABE also produces an optimal size ciphertext. Recently, such an optimal size CCA secure ABE has been constructed from WPRF and non-interactive zap [27] with a motivation to get a multi-attribute fully homomorphic encryption scheme. On the other hand, our ABE (described in App. D.2) is much simpler and relies solely on our CWPRF and a psudorandom generator (PRG).

Related Works. Realizing CCA security has been one of the primary goals of the research community after CPA security is confirmed for a particular primitive. For instance, a variety of IND-CCA secure PKEs [13,12,30,22,23,33,20] are proposed starting from a IND-CPA secure PKEs. Some of these techniques have been translated to design more advanced IND-CCA secure encryption such as ABE and PE. Goyal et al. [17] extended the procedure given by Canetti, Halevi, and Katz [12], of achieving any IND-CCA secure PKE from IND-CCA secure IBE, in case of key-policy ABE where they required that the IND-CPA secure ABE must satisfy delegatability. In a subsequent work, Yamada et al. [32] proposed generic transformations which take advantage of certain delegatability and verifiability properties of existing IND-CPA secure ABEs to convert these into IND-CCA secure schemes. To make a larger class of encryption schemes IND-CCA secure, Nandi and Pandit [24] extended the framework of [32] by introducing a weak version of delegatability and verifiability that must be present in the original schemes. Recently, Koppula and Waters [21] presented a black box transformation for IND-CCA security of any ABE or (one-sided) PE utilizing a hinting PRG along with the IND-CPA security of the considered primitive.

Blömer and Liske [5] showed a non-generic IND-CCA secure PE using the methodology of well-formedness proofs. Benhamouda et al. [3] constructed IND-CCA secure FEs for linear functionality from projective hash functions with homomorphic properties. Very recently, Pal and Dutta [27] directly built IND-CCA secure IBE and ABE for all circuits using WPRF. It can be noted that the

focus of all previous works was to consider a particular function class and depict IND-CCA security for FE associated to that class. On the contrary, we describe a IND-CCA secure PKFE for all circuits via a new cryptographic tool that we believe to have more potential applications in other aspects of cryptography.

2 Preliminaries

Notations. We denote the security parameter by λ, a natural number. For $m \in \{0,1\}^*$, $|m|$ indicated the size of the string m. To denote the sampling of an element s uniformly at random from a set $\mathcal{S}$, we use the notation $s \leftarrow \mathcal{S}$. For any probabilistic polynomial time (PPT) algorithm A, we define $r \leftarrow \mathsf{A}(s)$ to denote the process of computing r by executing A with an input s using a fresh randomness (unless A is a deterministic algorithm). All circuits and functions are of polynomial size. We say *negl* is a negligible function in an input parameter λ, if for all $c > 0$, there exists λ_0 such that $negl(\lambda) < \lambda^{-c}$ for all $\lambda > \lambda_0$.

2.1 Pseudorandom Generator

Definition 1 [Pseudorandom Generator] A pseudorandom generator (PRG) is a deterministic polynomial time algorithm PRG that on input a seed $s \in \{0,1\}^\lambda$ outputs a string of length $\ell(\lambda)$ such that the following holds:

- *output expansion*: We require $\ell(\lambda) > \lambda$ for all λ.
- *pseudorandomness security*: For all PPT adversary $\mathcal{A}$ and $s \leftarrow \{0,1\}^\lambda, r \leftarrow \{0,1\}^{\ell(\lambda)}$ the following advantage

$$\mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{A}}(\lambda) = |\Pr[\mathcal{A}(1^\lambda, \mathsf{PRG}(s)) = 1] - \Pr[\mathcal{A}(1^\lambda, r) = 1]\ |$$

  is a negligible function of λ.

2.2 Constrained Pseudorandom Function

Definition 2 [Constrained Pseudorandom Function] A constrained pseudorandom function (CPRF) is defined for a circuit class $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ and a domain $\mathcal{X}$. It consists of four PPT algorithms Setup, ConKey, $\mathsf{ConEval}$, Eval that work as follows:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{msk}$: The setup algorithm outputs a master secret-key msk.
- $\mathsf{ConKey}(\mathsf{msk}, C) \to \mathsf{sk}_C$: The constrained key algorithm generates a constrained key sk_C corresponding to a circuit $C \in \mathcal{C}_\lambda$.
- $\mathsf{ConEval}(\mathsf{sk}_C, x) \to y$: The constrained evaluation algorithm outputs a value $y \in \mathcal{Y}$ using the constrained sk_C for an input $x \in \mathcal{X}$.
- $\mathsf{Eval}(\mathsf{msk}, x) \to y$: The evaluation algorithm outputs an element $y \in \mathcal{Y}$ using the master secret-key msk for a string $x \in \mathcal{X}$.

A CPRF must satisfy the following requirements.

Definition 3 (Correctness) For all $\lambda \in \mathbb{N}, \mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda), x \in \mathcal{X}, C \in \mathcal{C}_\lambda$, and $\mathsf{sk}_C \leftarrow \mathsf{ConKey}(\mathsf{msk}, C)$ we have

correctness of ConEval. $\Pr[\mathsf{ConEval}(\mathsf{sk}_C, x) = \mathsf{Eval}(\mathsf{msk}, x) \text{ s.t. } C(x) = 1] = 1$

Definition 4 [Adaptive Pseudorandomness] For a security parameter $\lambda \in \mathbb{N}$, a circuit class $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ and a bit $b \in \{0,1\}$, we define the experiment $\mathsf{Adp\text{-}IND}_{\mathcal{A}}^{\mathsf{CPRF}}(1^\lambda, b)$ between a challenger and a PPT adversary $\mathcal{A}$ in the following manner:
Setup: The challenger runs $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$ and prepares two empty sets Q_c and Q_x.

Queries: After setup, $\mathcal{A}$ can make queries to the following oracles at any point of the experiment.

- *constrained key queries.* On input a circuit $C \in \mathcal{C}_\lambda$, the challenger returns a constrained key $\mathsf{sk}_C \leftarrow \mathsf{ConKey}(\mathsf{msk}, C)$ and updates the set $Q_c \leftarrow Q_c \cup \{C\}$.
- *evaluation queries.* On input $x \in \mathcal{X}$, the challenger returns a pseudorandom value $y \leftarrow \mathsf{Eval}(\mathsf{msk}, x)$ and updates the set $Q_x \leftarrow Q_x \cup \{x\}$.

Challenge: At some point, $\mathcal{A}$ submits a challenge string $x^* \in \mathcal{X}$. If $b = 0$, the challenger returns $y \leftarrow \mathsf{Eval}(\mathsf{msk}, x^*)$ to $\mathcal{A}$, otherwise it returns $y \leftarrow \mathcal{Y}$.

Guess: Eventually, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. The challenger returns 1 if $b = b'$, $x^* \notin Q_x$ and $C(x^*) = 0$ for all $C \in Q_c$.

A CPRF is said to be adaptively secure if the following advantage is a negligible function of λ:

$$\mathsf{Adv}^{\mathsf{Adp\text{-}IND}}_{\mathcal{A},\mathsf{CPRF}}(\lambda) = |\Pr[\mathsf{Adp\text{-}IND}^{\mathsf{CPRF}}_{\mathcal{A}}(1^\lambda, b) = 1] - \tfrac{1}{2}|$$

Definition 5 [Selective Pseudorandomness] We define a security experiment $\mathsf{Sel\text{-}IND}^{\mathsf{CPRF}}_{\mathcal{A}}(1^\lambda, b)$ for selective security of CPRF similarly to the experiment $\mathsf{Adp\text{-}IND}^{\mathsf{CPRF}}_{\mathcal{A}}(1^\lambda, b)$ except that the adversary $\mathcal{A}$ submits the challenge statement $x^* \in \mathcal{X}$ before any oracle query. We define the advantage $\mathsf{Adv}^{\mathsf{Sel\text{-}IND}}_{\mathcal{A},\mathsf{CPRF}}(\lambda)$ accordingly and say that the CPRF is selectively secure if the quantity is a negligible function of λ.

Definition 6 [Function Privacy] For a security parameter $\lambda \in \mathbb{N}$, a circuit class $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ and a bit $b \in \{0,1\}$, we define the experiment $\mathsf{FP\text{-}IND}^{\mathsf{CPRF}}_{\mathcal{A}}(1^\lambda, b)$ between a challenger and a PPT adversary $\mathcal{A}$ in the following manner:
Setup: The challenger runs $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$ and prepares two empty sets $Q_{c,c}$ and Q_x.

Queries: In any arbitrary order, $\mathcal{A}$ can make queries to the following oracles.

- *constrained key queries.* On input a pair of circuits $(C_0, C_1) \in \mathcal{C}_\lambda \times \mathcal{C}_\lambda$, the challenger returns a constrained key $\mathsf{sk}_{C_b} \leftarrow \mathsf{ConKey}(\mathsf{msk}, C_b)$ and updates the set $Q_{c,c} \leftarrow Q_{c,c} \cup \{(C_0, C_1)\}$.
- *evaluation queries.* On input $x \in \mathcal{X}$, the challenger returns a pseudorandom value $y \leftarrow \mathsf{Eval}(\mathsf{msk}, x)$ and updates the set $Q_x \leftarrow Q_x \cup \{x\}$.

Guess: Finally, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. The challenger returns 1 if $b = b'$ and the following conditions hold:

1. $C_0(x) = C_1(x)$ for all $(C_0, C_1) \in Q_{c,c}$ and $x \in Q_x$,
2. $S(C_0) \cap S(C_0') = S(C_1) \cap S(C_1')$ for any two distinct pairs $(C_0, C_1), (C_0', C_1')$ of $Q_{c,c}$ where $S(C) = \{x \in \mathcal{X} : C(x) = 1\}$

A CPRF is said to be function private if the following advantage is a negligible function of λ:

$$\mathsf{Adv}^{\mathsf{FP\text{-}IND}}_{\mathcal{A},\mathsf{CPRF}}(\lambda) = |\Pr[\mathsf{FP\text{-}IND}^{\mathsf{CPRF}}_{\mathcal{A}}(1^\lambda, b) = 1] - \tfrac{1}{2}|$$

The restrictions on constrained key queries in Def. 6 is necessary to prevent $\mathcal{A}$ in trivially distinguishing the keys sk_{C_0} and sk_{C_1}. This has been discussed in [8] with several examples. We define a weaker version of function privacy where an adversary is restricted to submit pair of circuits (C_0, C_1) such that $C_0(x) = C_1(x)$ for all $x \in \mathcal{X}$. Hence, the above two conditions are not needed in this case. We call this notion as *weak* function privacy. It is trivial to verify that a function private CPRF (Def. 6) is also weak function private.

Definition 7 [Weak Function Privacy] We define a security experiment $\mathsf{wFP\text{-}IND}_{\mathcal{A}}^{\mathsf{CPRF}}(1^\lambda, b)$ for weak function privacy security of CPRF similarly to the experiment $\mathsf{FP\text{-}IND}_{\mathcal{A}}^{\mathsf{CPRF}}(1^\lambda, b)$ except that all the constrained key queries $\{(C_0, C_1)\}$ of the adversary $\mathcal{A}$ must satisfy the condition that $C_0(x) = C_1(x)$ for all $x \in \mathcal{X}$ and the challenger returns 1 only if $b = b'$ in the guess step. We define the advantage $\mathsf{Adv}^{\mathsf{wFP\text{-}IND}}_{\mathcal{A},\mathsf{CPRF}}(\lambda)$ accordingly and say that the CPRF is weak function private if the quantity is a negligible function of λ.

2.3 Constrained Witness Pseudorandom Functions

Definition 8 [Constrained Witness Pseudorandom Functions] A constrained witness pseudorandom function (CWPRF) is defined for a circuit class $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ and an NP language L with a relation $R : \mathcal{X} \times \mathcal{W} \to \{0, 1\}$. It consists of five PPT algorithms Gen, ConKey, F, ConF, Eval that work as follows:

- $\mathsf{Gen}(1^\lambda, R) \to (\mathsf{fk}, \mathsf{ek})$: The generation algorithm outputs a secret function key fk and a public evaluation key ek.
- $\mathsf{ConKey}(\mathsf{fk}, C) \to \mathsf{fk}_C$: The constrained key algorithm generates a constrained key fk_C corresponding to a circuit $C \in \mathcal{C}_\lambda$.
- $\mathsf{F}(\mathsf{fk}, x) \to y$: The pseudorandom function outputs an element $y \in \mathcal{Y}$ using the secret function key fk for a string $x \in \mathcal{X}$.
- $\mathsf{ConF}(\mathsf{fk}_C, x) \to y$: The constrained pseudorandom function algorithm outputs a value $y \in \mathcal{Y}$utilizing the constrained key fk_C for an input $x \in \mathcal{X}$.
- $\mathsf{Eval}(\mathsf{ek}, x, w) \to y$: The evaluation algorithm outputs a value $y \in \mathcal{Y}$ using the public evaluation key ek and a witness $w \in \mathcal{W}$ for a string $x \in \mathcal{X}$.

A CWPRF must satisfy the following requirements.

Definition 9 (Correctness) For all $\lambda \in \mathbb{N}, (\mathsf{fk}, \mathsf{ek}) \leftarrow \mathsf{Gen}(1^\lambda, R), x \in \mathcal{X}, w \in \mathcal{W}, C \in \mathcal{C}_\lambda$, and $\mathsf{fk}_C \leftarrow \mathsf{ConKey}(\mathsf{fk}, C)$ we have

- *correctness of ConF.* $\Pr[\mathsf{ConF}(\mathsf{fk}_C, x) = \mathsf{F}(\mathsf{fk}, x) \text{ s.t. } C(x) = 1] = 1$
- *correctness of Eval.* $\Pr[\mathsf{Eval}(\mathsf{ek}, x, w) = \mathsf{F}(\mathsf{fk}, x) \text{ s.t. } R(x, w) = 1] = 1$

Definition 10 [Adaptive Pseudorandomness] For a security parameter $\lambda \in \mathbb{N}$, a circuit class $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$, an NP language L with a relation $R : \mathcal{X} \times \mathcal{W} \to \{0, 1\}$ and a bit $b \in \{0, 1\}$, we define the experiment $\mathsf{Adp\text{-}IND}_{\mathcal{A}}^{\mathsf{CWPRF}}(1^\lambda, b)$ between a challenger and a PPT adversary $\mathcal{A}$ in the following manner:

Setup: The challenger runs $(\mathsf{fk}, \mathsf{ek}) \leftarrow \mathsf{Gen}(1^\lambda, R)$ and sends ek to $\mathcal{A}$. It also prepares two empty sets Q_c and Q_x.

Queries: After setup, $\mathcal{A}$ can make queries to the following oracles at any point of the experiment.

– *constrained key queries.* On input a circuit $C \in \mathcal{C}_\lambda$, the challenger returns a constrained key $\mathsf{fk}_C \leftarrow \mathsf{ConKey}(\mathsf{fk}, C)$ and updates the set $Q_c \leftarrow Q_c \cup \{C\}$.
– *pseudorandom function queries.* On input $x \in \mathcal{X}$, the challenger returns a pseudorandom value $y \leftarrow \mathsf{F}(\mathsf{fk}, x)$ and updates the set $Q_x \leftarrow Q_x \cup \{x\}$.

Challenge: At some point, $\mathcal{A}$ submits a challenge string $x^* \in \mathcal{X} \setminus L$. If $b = 0$, the challenger returns $y \leftarrow \mathsf{F}(\mathsf{fk}, x^*)$ to $\mathcal{A}$, otherwise it returns $y \leftarrow \mathcal{Y}$.

Guess: Eventually, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. The challenger returns 1 if $b = b'$, $x^* \notin Q_x$ and $C(x^*) = 0$ for all $C \in Q_c$.

A CWPRF is said to be adaptively secure if the following advantage is a negligible function of λ:

$$\mathsf{Adv}^{\mathsf{Adp\text{-}IND}}_{\mathcal{A},\mathsf{CWPRF}}(\lambda) = |\Pr[\mathsf{Adp\text{-}IND}^{\mathsf{CWPRF}}_{\mathcal{A}}(1^\lambda, b) = 1] - \tfrac{1}{2}|$$

Definition 11 [Selective Pseudorandomness] We define a security experiment $\mathsf{Sel\text{-}IND}^{\mathsf{CWPRF}}_{\mathcal{A}}(1^\lambda, b)$ for selective security of CWPRF similarly to the experiment $\mathsf{Adp\text{-}IND}^{\mathsf{CWPRF}}_{\mathcal{A}}(1^\lambda, b)$ except that the adversary $\mathcal{A}$ submits the challenge statement $x^* \in \mathcal{X} \setminus L$ before setup phase. We define the advantage $\mathsf{Adv}^{\mathsf{Sel\text{-}IND}}_{\mathcal{A},\mathsf{CWPRF}}(\lambda)$ accordingly and say that the CWPRF is selectively secure if the quantity is a negligible function of λ.

Definition 12 [Function Privacy] For a security parameter $\lambda \in \mathbb{N}$, a circuit class $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$, an NP language L with a relation $R : \mathcal{X} \times \mathcal{W} \to \{0,1\}$ and a bit $b \in \{0,1\}$, we define the experiment $\mathsf{FP\text{-}IND}^{\mathsf{CWPRF}}_{\mathcal{A}}(1^\lambda, b)$ between a challenger and a PPT adversary $\mathcal{A}$ in the following manner:
Setup: The challenger runs $(\mathsf{fk}, \mathsf{ek}) \leftarrow \mathsf{Gen}(1^\lambda, R)$ and sends ek to $\mathcal{A}$. It also prepares two empty sets $Q_{c,c}$ and Q_x.

Queries: $\mathcal{A}$ can make queries to the following oracles in any arbitrary order.
– *constrained key queries.* On input a pair of circuits $(C_0, C_1) \in \mathcal{C}_\lambda \times \mathcal{C}_\lambda$, the challenger returns a constrained key $\mathsf{fk}_{C_b} \leftarrow \mathsf{ConKey}(\mathsf{fk}, C_b)$ and updates the set $Q_{c,c} \leftarrow Q_{c,c} \cup \{(C_0, C_1)\}$.
– *pseudorandom function queries.* On input $x \in \mathcal{X}$, the challenger returns a pseudorandom value $y \leftarrow \mathsf{F}(\mathsf{fk}, x)$ and updates the set $Q_x \leftarrow Q_x \cup \{x\}$.

Guess: Finally, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. The challenger returns 1 if $b = b'$ and the following conditions hold:
1. $C_0(x) = C_1(x)$ for all $(C_0, C_1) \in Q_{c,c}$ and $x \in Q_x$,
2. $C_0(x) = C_1(x)$ for all $(C_0, C_1) \in Q_{c,c}$ and $x \in L$,
3. $S(C_0) \cap S(C'_0) = S(C_1) \cap S(C'_1)$ for any two distinct pairs $(C_0, C_1), (C'_0, C'_1)$ of $Q_{c,c}$ where $S(C) = \{x \in \mathcal{X} : C(x) = 1\}$

A CWPRF is said to be function private if the following advantage is a negligible function of λ:

$$\mathsf{Adv}^{\mathsf{FP\text{-}IND}}_{\mathcal{A},\mathsf{CWPRF}}(\lambda) = |\Pr[\mathsf{FP\text{-}IND}^{\mathsf{CWPRF}}_{\mathcal{A}}(1^\lambda, b) = 1] - \tfrac{1}{2}|$$

Definition 13 [Weak Function Privacy] We define a security experiment $\mathsf{wFP\text{-}IND}^{\mathsf{CWPRF}}_{\mathcal{A}}(1^\lambda, b)$ for weak function privacy security of CWPRF similarly to the experiment $\mathsf{FP\text{-}IND}^{\mathsf{CWPRF}}_{\mathcal{A}}(1^\lambda, b)$ except that all the constrained key queries

$\{(C_0, C_1)\}$ of the adversary $\mathcal{A}$ must satisfy the condition that $C_0(x) = C_1(x)$ for all $x \in \mathcal{X}$ and the challenger returns 1 only if $b = b'$ in the guess step. We define the advantage $\mathsf{Adv}^{\mathsf{wFP\text{-}IND}}_{\mathcal{A},\mathsf{CWPRF}}(\lambda)$ accordingly and say that the CWPRF is weak function private if the quantity is a negligible function of λ.

2.4 Puncturable Witness Pseudorandom Function

A puncturable WPRF (PWPRF) [28] is a special case of CWPRF where the constrained keys are generated only for the point circuits, that is the circuit class $\{C_\lambda\}_{\lambda\in\mathbb{N}}$ contains circuits of the form $C_{x'}$ for a particular point $x' \in \mathcal{X}$ and $C_{x'}(x) = 1$ if and only if $x \neq x'$. Specifically, a PWPRF for an NP language L with a relation $R : \mathcal{X} \times \mathcal{W} \to \{0, 1\}$ is defined by a set of five PPT algorithms Gen, PuncKey, F, PuncF, Eval which work in an identical way as regular CWPRF except that the PuncKey algorithm takes in a string $x \in \mathcal{X}$ instead of a circuit C. For correctness, we require that for all $\lambda \in \mathbb{N}, (\mathsf{fk}, \mathsf{ek}) \leftarrow \mathsf{Gen}(1^\lambda, R), x \in \mathcal{X}, w \in \mathcal{W}, C \in \mathcal{C}_\lambda$, and $\mathsf{fk}_{x'} \leftarrow \mathsf{PuncKey}(\mathsf{fk}, x')$ we have

- *correctness of PuncF.* $\Pr[\mathsf{PuncF}(\mathsf{fk}_{x'}, x) = \mathsf{F}(\mathsf{fk}, x) \text{ s.t. } x \neq x'] = 1$
- *correctness of Eval.* $\Pr[\mathsf{Eval}(\mathsf{ek}, x, w) = \mathsf{F}(\mathsf{fk}, x) \text{ s.t. } R(x, w) = 1] = 1$

We can define the Adp-IND and Sel-IND security notions of PWPRF in a similar fashion as we have described the security for CWPRF in Def. 10 and Def. 11 (Sec. 3). The pseudorandomness of PWPRF states that the value $\mathsf{F}(\mathsf{fk}, x^*)$ remains pseudorandom even when an adversary gets a punctured key fk_{x^*} where x^* is the challenge statement lying outside the language L. A formal description of the security notions is provided in App. A.

2.5 Functional Encryption

Definition 14 [Public-key Functional Encryption] A public-key functional encryption (PKFE) is defined for a class of functions $\{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ and a message space $\mathcal{M}$. It consists of four PPT algorithms Setup, KeyGen, Enc, Dec that work as follows:

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{msk}, \mathsf{pp})$: The Setup algorithm outputs a master secret-key msk and a public parameter pp.
- $\mathsf{KeyGen}(\mathsf{msk}, f) \to \mathsf{sk}_f$: The key generation algorithm generates a secret-key sk_f corresponding to a function $f \in \mathcal{F}_\lambda$, and outputs .
- $\mathsf{Enc}(\mathsf{pp}, m) \to \mathsf{ct}$: The encryption algorithm outputs a ciphertext ct by encrypting a message $m \in \mathcal{M}$ using the public parameter pp.
- $\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}) \to y$: The decryption algorithm decrypts the ciphertext ct using the secret-key sk_f and outputs a value y.

A PKFE must satisfy the following requirements.

Definition 15 (Correctness) For all $\lambda \in \mathbb{N}, (\mathsf{msk}, \mathsf{pp}) \leftarrow \mathsf{Setup}(1^\lambda), m \in \mathcal{M}$, $f \in \mathcal{F}_\lambda$ and $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$ we have

- *correctness of Dec.* $\Pr[\mathsf{Dec}(\mathsf{sk}_f, \mathsf{Enc}(\mathsf{pp}, m)) = f(m)] = 1 - \mathsf{negl}(\lambda)$

Definition 16 [Adaptive Indistinguishability CCA security] For a security parameter $\lambda \in \mathbb{N}$, a function class $\{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$, a message space $\mathcal{M}$ and a bit $b \in \{0, 1\}$,

we define the experiment $\mathsf{Adp\text{-}INDCCA}_{\mathcal{A}}^{\mathsf{PKFE}}(1^\lambda, b)$ between a challenger and a PPT adversary $\mathcal{A}$ in the following manner:

Setup: The challenger runs $(\mathsf{msk}, \mathsf{pp}) \leftarrow \mathsf{Setup}(1^\lambda)$ and sends pp to $\mathcal{A}$. It also prepares two empty sets Q_f and $Q_{ct,f}$.

Queries: After setup, $\mathcal{A}$ can query to the following oracles at any point of the experiment.

- *secret-key queries.* On input a function $f \in \mathcal{F}_\lambda$, the challenger returns a secret-key $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$ and updates the set $Q_f \leftarrow Q_f \cup \{f\}$.
- *decryption queries.* On input a ciphertext, function pair (ct, f), the challenger computes $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$ and returns $\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct})$. It also updates $Q_{ct,f} \leftarrow Q_{ct,f} \cup \{(\mathsf{ct}, f)\}$

Challenge: At some point, $\mathcal{A}$ submits a pair of challenge messages $(m_0, m_1) \in \mathcal{M} \times \mathcal{M}$. The challenger returns $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pp}, m_b)$ to $\mathcal{A}$.

Guess: Eventually, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. The challenger returns 1 if $b = b'$ and $f(m_0) = f(m_1)$ holds for all $f \in Q_f$ and for all $(\mathsf{ct}^*, f) \in Q_{ct,f}$.

A PKFE is said to be adaptive indistinguishability CCA secure if the following advantage is a negligible function of λ:

$$\mathsf{Adv}_{\mathcal{A},\mathsf{PKFE}}^{\mathsf{Adp\text{-}INDCCA}}(\lambda) = |\Pr[\mathsf{Adp\text{-}INDCCA}_{\mathcal{A}}^{\mathsf{PKFE}}(1^\lambda, b) = 1] - \tfrac{1}{2}|$$

2.6 Indistinguishability Obfuscation

Definition 17 [Indistinguishability Obfuscation] An indistinguishability obfuscator for a class of circuits $\{\mathcal{C}_\lambda\}$ is a PPT algorithm $i\mathcal{O}$ which satisfies the following properties:

- *Functionality*: For all security parameter $\lambda \in \mathbb{N}$, for all $C \in \mathcal{C}_\lambda$, for all inputs x, we require that

$$\Pr[\widetilde{C}(x) = C(x) : \widetilde{C} \leftarrow i\mathcal{O}(1^\lambda, C)] = 1$$

- *Indistinguishability*: For any PPT distinguisher $\mathcal{D}$ and for all pair of circuits $C_0, C_1 \in \mathcal{C}_\lambda$ that compute the same function and are of same size, the following advantage is a negligible function of λ:

$$\mathsf{Adv}_{\mathcal{D}}^{i\mathcal{O}}(\lambda) = |\Pr[\mathcal{D}(\widetilde{C}, C_0, C_1) = b \text{ s.t. } \widetilde{C} \leftarrow i\mathcal{O}(1^\lambda, C_b), b \leftarrow \{0,1\}] - \tfrac{1}{2}|$$

3 Construction of CWPRF from CPRF and $i\mathcal{O}$

Our construction of CWPRF is inspired by the $i\mathcal{O}$ based PRF constructions of [8,26]. Specifically, we replace the PRF in the construction of [26] with a suitable CPRF that supports any polynomial size circuits. For constrain hiding, we require that the ConEval algorithm of the underlying CPRF to output pseudorandom values for inputs x such that $C(x) = 0$. This is due to the fact that if ConEval outputs $\perp$ on inputs where the circuit evaluates to zero then the constrained key sk_C reveals information about the circuit. One such CPRF is the $i\mathcal{O}$-based construction of [8] that we may choose to instantiate our CWPRF.

We build of a selectively secure $\mathsf{CWPRF} = (\mathsf{Gen}, \mathsf{ConKey}, \mathsf{F}, \mathsf{ConF}, \mathsf{Eval})$ for an NP language L with a witness relation $R : \mathcal{X} \times \mathcal{W} \to \{0,1\}$ using an indistinguishability obfuscator $i\mathcal{O}$ and a $\mathsf{CPRF} = (\mathsf{Setup}, \mathsf{ConKey}, \mathsf{ConEval}, \mathsf{Eval})$ for a class of circuits $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ and a domain $\mathcal{X}$. The CWPRF works as follows:

$\mathsf{CWPRF.Gen}(1^\lambda, R)$: It computes a master secret-key $\mathsf{msk} \leftarrow \mathsf{CPRF.Setup}(1^\lambda)$ and generate an obfuscated circuit $\widetilde{F} \leftarrow i\mathcal{O}(1^\lambda, F_{\mathsf{msk},R})$ where the circuit $F_{\mathsf{msk},R}$ is defined as follows:

$$F_{\mathsf{msk},R}(x,w) = \begin{cases} \mathsf{CPRF.Eval}(\mathsf{msk}, x) & \text{if } R(x,w) = 1 \\ \perp & \text{otherwise} \end{cases}$$

It then outputs the function secret-key as $\mathsf{fk} = \mathsf{msk}$ and the public evaluation key as $\mathsf{ek} = \widetilde{F}$.

$\mathsf{CWPRF.ConKey}(\mathsf{fk}, C)$: For a circuit $C \in \mathcal{C}_\lambda$, the constrained key algorithm uses $\mathsf{fk} = \mathsf{msk}$ and returns $\mathsf{fk}_C \leftarrow \mathsf{CPRF.ConKey}(\mathsf{msk}, C)$.

$\mathsf{CWPRF.ConF}(\mathsf{fk}_C, x)$: For any $x \in \mathcal{X}$, the constrained evaluation algorithm outputs $\mathsf{CPRF.ConEval}(\mathsf{fk}_C, x)$.

$\mathsf{CWPRF.F}(\mathsf{fk}, x)$: Using the function secret-key as $\mathsf{fk} = \mathsf{msk}$, it outputs the pseudorandom value corresponding to an $x \in \mathcal{X}$ as $\mathsf{CPRF.Eval}(\mathsf{msk}, x) \in \mathcal{Y}$.

$\mathsf{CWPRF.Eval}(\mathsf{ek}, x, w)$: It takes the evaluation key $\mathsf{ek} = \widetilde{F}$ and outputs $\widetilde{F}(x,w)$ for $x \in \mathcal{X}$ and $w \in \mathcal{W}$.

Correctness. The correctness of CWPRF.ConF algorithm directly follows from the correctness of CPRF.ConEval. For the correctness of CWPRF.Eval, we note that if w is a valid witness of the statement x then $F_{\mathsf{msk},R}(x,w) = \widetilde{F}(x,w) = \mathsf{CPRF.Eval}(\mathsf{msk}, x)$ holds by the correctness of $i\mathcal{O}$. Therefore, $\mathsf{CWPRF.Eval}(\mathsf{ek}, x, w) = \mathsf{CWPRF.F}(\mathsf{fk}, x)$ if $R(x,w) = 1$.

Theorem 1 *The constrained witness pseudorandom function CWPRF described above is Sel-IND secure (as per Def. 11) assuming the $i\mathcal{O}$ is a secure indistinguishability obfuscator (as per Def. 17) and the CPRF is Sel-IND secure (as per Def. 5).*

Theorem 2 *The constrained witness pseudorandom function CWPRF described above is wFP-IND secure (as per Def. 13) assuming the $i\mathcal{O}$ is a secure indistinguishability obfuscator (as per Def. 17) and the CPRF is wFP-IND secure (as per Def. 7).*

Proof Sketch. In the Sel-IND game of CWPRF, the challenger computes a circuit E_{x^*} which is satisfied by all inputs other than the challenge statement x^*. Now, it sets $\mathsf{sk}^* \leftarrow \mathsf{CPRF.ConKey}(\mathsf{msk}, E_{x^*})$ and defines $\mathsf{ek} = i\mathcal{O}(1^\lambda, \widetilde{F})$ where $\widetilde{F}(x,w) = \mathsf{CPRF.ConEval}(\mathsf{sk}^*, x)$ if $R(x,w) = 1$, 0 otherwise. The circuits $F_{\mathsf{msk},R}$ and $\widetilde{F}$ are equivalent since $x^* \notin L$. Therefore, an adversary cannot detect the change in the evaluation key ek by the security of $i\mathcal{O}$ and $\mathsf{CWPRF.F}(\mathsf{fk}, x^*)$ remains pseudorandom by the Sel-IND security of CPRF.

To show the wFP-IND security of CWPRF, we follow the similar technique as above. The challenger picks a random statement x^*, computes $r^* = \mathsf{CPRF.Eval}(\mathsf{msk}, x^*)$, $\mathsf{sk}^* \leftarrow \mathsf{CPRF.ConKey}(\mathsf{msk}, E_{x^*})$ and then sets $\mathsf{ek} = i\mathcal{O}(1^\lambda, \widetilde{F}_{x^*,r^*})$. The circuit $\widetilde{F}_{x^*,r^*}(x, w) = \mathsf{CPRF.ConEval}(\mathsf{sk}^*, x)$ if $R(x, w) = 1$ and returns r^* if $x = x^*$, 0 otherwise. The security of $i\mathcal{O}$ guarantees that this change in ek remains indistinguishable to an adversary. Finally, we can show that WCPRF satisfies wFP-IND security under the assumption of wFP-IND security of CPRF. We give formal proofs in App. B.

Remark 1 *The CPRF of Boneh et al. [8] requires that the underlying PRF and $i\mathcal{O}$ to be secure against subexponential adversaries (Theorem 3.3 of [8]) as their aim was to achieve (strong) function privacy (Def. 6). If the challenger circuits given by an adversary (in the security game of Def. 6) differs only on a polynomial number of points, then polynomial security of the PRF and $i\mathcal{O}$ suffices (Remaark D.11 of [8]). Since the weak function privacy (Def. 7) restricts the challenge circuits to be equivalent, we are able to base the security of CWPRF relying on CPRF and $i\mathcal{O}$ both secure against polynomial time adversaries.*

4 Construction of CCA secure PKFE from CWPRF

Our generic construction of PKFE only requires CWPRF along with a pseudorandom generator. Mainly, we translate the SKFE of Boneh et al. [8] in public-key setting and more importantly we achieve security against active adversaries. Formally, we build a $\mathsf{PKFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for all polynomial-size boolean functions having input space $\mathcal{M} = \{0,1\}^\ell$. Let us consider a length doubling PRG with domain $\{0,1\}^\lambda$ for some $\lambda \in \mathbb{N}$. We take a $\mathsf{PWPRF} = (\mathsf{Gen}, \mathsf{PuncKey}, \mathsf{F}, \mathsf{PuncF}, \mathsf{Eval})$[1] for the NP language $L = \{r \in \{0,1\}^{2\lambda} : \exists\ s \text{ s.t. } \mathsf{PRG}(s) = r\}$ with relation R and a $\mathsf{CWPRF} = (\mathsf{Gen}, \mathsf{ConKey}, \mathsf{F}, \mathsf{ConF}, \mathsf{Eval})$ for the NP language $L_{\mathsf{ek}} = \{(r, c) \in \{0,1\}^{2\lambda+\ell} : \exists\ (s, m) \text{ s.t. } c = m \oplus \mathsf{PWPRF.Eval}(\mathsf{ek}, r, s)\}$ with a relation R_{ek} where ek is an evaluation key of PWPRF. The PWPRF has a domain of size 2λ and the CWPRF has a domain of size $2\lambda + \ell$. We assume that the CWPRF is associated with a class of boolean functions taking inputs from $\{0,1\}^{2\lambda+\ell}$.

$\mathsf{PKFE.Setup}(1^\lambda)$: The setup algorithm proceeds as follows:

1. Generate a key pair $(\mathsf{fk}, \mathsf{ek}) \leftarrow \mathsf{PWPRF.Gen}(1^\lambda, R)$ for a relation R defined by the language L as above.
2. Define a language L_{ek} with a relation R_{ek} as above and generate a key pair $(\mathsf{fk}', \mathsf{ek}') \leftarrow \mathsf{CWPRF.Gen}(1^\lambda, R_{\mathsf{ek}})$.
3. Return the master secret-key as $\mathsf{msk} = (\mathsf{fk}, \mathsf{fk}')$ and the public parameter as $\mathsf{pp} = (\mathsf{ek}, \mathsf{ek}')$.

$\mathsf{PKFE.KeyGen}(\mathsf{msk}, f)$: The key generation algorithm produces a secret-key corresponding to the boolean function f with input length ℓ as follows:

1. Parse $\mathsf{msk} = (\mathsf{fk}, \mathsf{fk}')$.

[1] We assume that the co-domain of the pseudorandom function is $\{0,1\}^\ell$ [28].

2. Define a circuit $C_{f,\mathsf{fk}}$ with constants f and fk as

$$C_{f,\mathsf{fk}}(r,c) = \begin{cases} 1 & \text{if } f(\mathsf{PWPRF.F}(\mathsf{fk},r) \oplus c) = 1 \\ 0 & \text{otherwise} \end{cases}$$

3. Compute the constrained key $\mathsf{fk}'_{C_{f,\mathsf{fk}}} \leftarrow \mathsf{CWPRF.ConKey}(\mathsf{fk}', C_{f,\mathsf{fk}})$.
4. Return the secret-key $\mathsf{sk}_f = \mathsf{fk}'_{C_{f,\mathsf{fk}}}$.

$\mathsf{PKFE.Enc}(\mathsf{pp}, m)$: The encryption algorithm computes a ciphertext for the message $m \in \{0,1\}^\ell$ as follows:

1. Parse $\mathsf{pp} = (\mathsf{ek}, \mathsf{ek}')$.
2. Pick a random string $s \leftarrow \{0,1\}^\lambda$ and set $r = \mathsf{PRG}(s)$.
3. Use PWPRF to compute $c = m \oplus \mathsf{PWPRF.Eval}(\mathsf{ek}, r, s)$ where s acts like a witness for the statement r of the language L.
4. Set a statement (r,c) of the language L_{ek} with a witness (s,m) and generate a pseudorandom value $v = \mathsf{CWPRF.Eval}(\mathsf{ek}', (r,c), (s,m))$.
5. Return the ciphertext as $\mathsf{ct} = (r,c,v)$.

$\mathsf{PKFE.Dec}(\mathsf{sk}_f, \mathsf{ct})$: The decryption algorithm proceeds as follows:

1. Parse $\mathsf{sk}_f = \mathsf{fk}'_{C_{f,\mathsf{fk}}}$ and $\mathsf{ct} = (r,c,v)$.
2. Extract the statement (r,c) of the language L_{ek} from the ciphertext ct and compute a pseudorandom value $v' = \mathsf{CWPRF.ConF}(\mathsf{fk}_{C_{f,\mathsf{fk}}}, (r,c))$.
3. Return 1 if $v = v'$ and 0 otherwise.

Correctness. Let $\mathsf{sk}_f = \mathsf{fk}'_{C_{f,\mathsf{fk}}}$ be a secret-key corresponding to a function f and $\mathsf{ct} = (r,c,v)$ be a ciphertext encrypting a message m. To show the correctness of decryption algorithm, we first note that if $C_{f,\mathsf{fk}}(r,c) = 1$ then by the correctness of $\mathsf{CWPRF.ConF}$ (Def. 8), it holds with probability 1 that $\mathsf{CWPRF.ConF}(\mathsf{fk}'_{C_{f,\mathsf{fk}}}, (r,c)) = \mathsf{CWPRF.F}(\mathsf{fk}', (r,c)) = \mathsf{CWPRF.Eval}(\mathsf{ek}', (r,c), (s, m)) = v$. Now, by the definition of the circuit $C_{f,\mathsf{fk}}$ and correctness of $\mathsf{PWPRF.Eval}$, we have $C_{f,\mathsf{fk}}(r,c) = 1$ holds if

$$1 = f(\mathsf{PWPRF.F}(\mathsf{fk},r) \oplus c) = f(\mathsf{PWPRF.Eval}(\mathsf{ek},r,s) \oplus c) = f(m)$$

since $c = \mathsf{PWPRF.Eval}(\mathsf{ek}, r, s) \oplus m$. Hence, the decryption successfully returns $f(m) = 1$ by checking $\mathsf{CWPRF.ConF}(\mathsf{fk}'_{C_{f,\mathsf{fk}}}, (r,c)) = v$. On the other hand, the correctness of $\mathsf{CWPRF.Eval}$ (Def. 8) and the pseudorandomness property of CWPRF (Def. 11) together implies that

$$v = \mathsf{CWPRF.Eval}(\mathsf{ek}', (r,c), (s,m)) = \mathsf{CWPRF.F}(\mathsf{fk}', (r,c))$$

remains pseudorandom with overwhelming probability if we have a constrained key $\mathsf{fk}'_{C_{f,\mathsf{fk}}}$ such that $C_{f,\mathsf{fk}}(r,c) = 0$. Hence, the decryption returns $f(m) = 0$ with high probability by checking $\mathsf{CWPRF.ConF}(\mathsf{fk}'_{C_{f,\mathsf{fk}}}, (r,c)) \neq v$.

Succinctness. An FE scheme is said to be succinct [16] if the size of the ciphertext is independent of the size of the computing function and may grow with the depth of the function. In our PKFE construction above, the size of a ciphertext $\mathsf{ct} = (r,c,v)$ encrypting a message $m \in \{0,1\}^\ell$ is given by $|\mathsf{ct}| = |r| + |c| + |v| =$

$2\lambda + \ell + \text{poly}(\lambda, \ell) = |m| + \text{poly}(\lambda)$ where we assume that ℓ is a polynomial in the security parameter λ. The ciphertext is completely independent of the size of the computing function. Therefore, our PKFE produces not only succinct ciphertexts, the size is optimal for any public-key system. On the other hand, the $i\mathcal{O}$-based PKFE of [15] uses fully homomorphic encryption (FHE) to encrypt messages and a NIZK proof system to prove a the well-formedness of FHE ciphertexts. Thus, the ciphertext size of [15] is not optimal and the ciphertext overhead is also huge in comparison to the ciphertext of our PKFE.

Theorem 3 *The public-key functional encryption scheme PKFE described above is Adp-INDCCA secure (as per Def. 16) assuming the PRG is secure (as per Def. 1), the PWPRF is Sel-IND secure (as per Def. 19) and the CWPRF is wFP-IND secure (as per Def. 13).*

Proof. The security analysis involves a sequence of hybrid experiments and proving indistinguishability between the experiments. The main idea of the proof is to mask the challenge message with a pseudorandom value corresponding to a statement of PWPRF that does not have any witness. Therefore, the evaluation key of PWPRF will not help any PPT adversary $\mathcal{A}$ to learn anything about the challenge message by the Sel-IND security of PWPRF. However, the main challenge comes into simulating the queries of the active adversary where we need to utilize the wFP-IND security of CWPRF. Let H_i be the event that the challenger outputs 1 in the i-th hybrid experiment.

$\mathbf{Hybd}_0$: We describe the hybrid 0 which is the standard $\mathsf{Adp\text{-}INDCCA}(1^\lambda, b)$ experiment as described in Def. 16:

Setup: The challenger generates two pair of keys $(\mathsf{fk}, \mathsf{ek}) \leftarrow \mathsf{PWPRF.Gen}(1^\lambda, R)$, $(\mathsf{fk}', \mathsf{ek}') \leftarrow \mathsf{CWPRF.Gen}(1^\lambda, R_{\mathsf{ek}})$ and sends $\mathsf{pp} = (\mathsf{ek}, \mathsf{ek}')$ to $\mathcal{A}$. It creates two empty sets Q_f and $Q_{ct,f}$.

Queries: $\mathcal{A}$ can query to the following oracles at any point of the experiment.

- *secret-key queries.* On input a function $f \in \mathcal{F}_\lambda$, the challenger computes $\mathsf{fk}'_{C_{f,\mathsf{fk}}} \leftarrow \mathsf{CWPRF.ConKey}(\mathsf{fk}', C_{f,\mathsf{fk}})$ and returns $\mathsf{sk}_f = \mathsf{fk}'_{C_{f,\mathsf{fk}}}$. It updates the set $Q_f \leftarrow Q_f \cup \{f\}$.
- *decryption queries.* On input a ciphertext-function pair (ct, f), the challenger parses $\mathsf{ct} = (r, c, v)$. It computes $\mathsf{fk}'_{C_{f,\mathsf{fk}}} \leftarrow \mathsf{CWPRF.ConKey}(\mathsf{fk}', C_{f,\mathsf{fk}})$ and sets $v' = \mathsf{CWPRF.ConF}(\mathsf{fk}'_{C_{f,\mathsf{fk}}}, (r, c))$. It outputs 1 if $v' = v$, 0 otherwise. It updates $Q_{ct,f} \leftarrow Q_{ct,f} \cup (\mathsf{ct}, f)$

Challenge: The adversary $\mathcal{A}$ submits a pair of challenge messages $(m_0, m_1) \in \{0,1\}^\ell \times \{0,1\}^\ell$. The challenger proceeds as follows:

1. Pick $s^* \leftarrow \{0,1\}^\lambda$ and set $r^* = \mathsf{PRG}(s^*)$.
2. Mask the challenge message as $c^* = m_b \oplus \mathsf{PWPRF.Eval}(\mathsf{ek}, r^*, s^*)$.
3. Compute a pseudorandom value $v^* = \mathsf{CWPRF.Eval}(\mathsf{ek}', (r^*, c^*), (s^*, m_b))$.
4. Return the challenge ciphertext $\mathsf{ct}^* = (r^*, c^*, v^*)$ to $\mathcal{A}$.

Guess: The adversary $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. The challenger returns 1 if $b = b'$ and $f(m_0) = f(m_1)$ holds for all $f \in Q_f$ and for all $(\mathsf{ct}^*, f) \in Q_{ct,f}$.

Hybd$_1$: It is exactly the same as hybrid 0 except that the challenger uses the master secret-key $\mathsf{msk} = (\mathsf{fk}, \mathsf{fk}')$ to compute ct^* as follows:

Challenge: The adversary $\mathcal{A}$ submits a pair of challenge messages $(m_0, m_1) \in \{0,1\}^\ell \times \{0,1\}^\ell$. The challenger proceeds as follows:

1. Pick $s^* \leftarrow \{0,1\}^\lambda$ and set $r^* = \mathsf{PRG}(s^*)$.
2. Mask the challenge message as $c^* = m_b \oplus \mathsf{PWPRF.F}(\mathsf{fk}, r^*)$.
3. Compute a pseudorandom value $v^* = \mathsf{CWPRF.F}(\mathsf{fk}', (r^*, c^*))$.
4. Return the challenge ciphertext $\mathsf{ct}^* = (r^*, c^*, v^*)$ to $\mathcal{A}$.

We note that the ciphertext distributions in both the hybrids are identical since by the correctness of $\mathsf{PWPRF.Eval}$ and $\mathsf{CWPRF.Eval}$ we have

$$\mathsf{PWPRF.Eval}(\mathsf{ek}, r^*, s^*) = \mathsf{PWPRF.F}(\mathsf{fk}, r^*) \text{ and}$$

$$\mathsf{CWPRF.Eval}(\mathsf{ek}', (r^*, c^*), (s^*, m_b)) = \mathsf{CWPRF.F}(\mathsf{fk}', (r^*, c^*)).$$

Therefore, Hybd_0 and Hybd_1 are identically distributed from $\mathcal{A}$'s view and we have $\Pr[\mathsf{H}_0] = \Pr[\mathsf{H}_1]$.

Hybd$_2$: In this hybrid, the challenger chooses r^* uniformly at random from $\{0,1\}^{2\lambda}$ instead of computing it as $r^* = \mathsf{PRG}(s^*)$ for some $s^* \in \{0,1\}^\lambda$. The rest of the experiment is same as Hybd_1. We indicate the change below.

Challenge: The adversary $\mathcal{A}$ submits a pair of challenge messages $(m_0, m_1) \in \{0,1\}^\ell \times \{0,1\}^\ell$. The challenger proceeds as follows:

1. Set $r^* \leftarrow \{0,1\}^{2\lambda}$.
2. Mask the challenge message as $c^* = m_b \oplus \mathsf{PWPRF.F}(\mathsf{fk}, r^*)$.
3. Compute a pseudorandom value $v^* = \mathsf{CWPRF.F}(\mathsf{fk}', (r^*, c^*))$.
4. Return the challenge ciphertext $\mathsf{ct}^* = (r^*, c^*, v^*)$ to $\mathcal{A}$.

The security of PRG (Def. 1) implies $|\Pr[\mathsf{H}_1] - \Pr[\mathsf{H}_2]| = \mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{B}_1}(\lambda) = negl(\lambda)$.

Hybd$_3$: The challenger modifies hybrid 2 using a punctured key which allows it to avoid the secret function key fk during the secret-key and decryption queries. We describe this hybrid as follows:

Setup: The challenger generates two pair of keys $(\mathsf{fk}, \mathsf{ek}) \leftarrow \mathsf{PWPRF.Gen}(1^\lambda, R)$, $(\mathsf{fk}', \mathsf{ek}') \leftarrow \mathsf{CWPRF.Gen}(1^\lambda, R_{\mathsf{ek}})$ and sends $\mathsf{pp} = (\mathsf{ek}, \mathsf{ek}')$ to $\mathcal{A}$. It creates two empty sets Q_f and $Q_{ct,f}$. Next, the challenger picks $r^* \leftarrow \{0,1\}^{2\lambda}$ (to be used in the challenge phase to mask m_b) in advance and computes a punctured key $\mathsf{fk}_{r^*} \leftarrow \mathsf{PWPRF.PuncKey}(\mathsf{fk}, r^*)$ and the pseudorandom value $u^* \leftarrow \mathsf{PWPRF.F}(\mathsf{fk}, r^*)$ in the setup itself.

Queries: $\mathcal{A}$ can query to the following oracles at any point of the experiment.

– *secret-key queries.* On input a function $f \in \mathcal{F}_\lambda$, the challenger defines the circuit

$$C_{f,\mathsf{fk}_{r^*},u^*}(r, c) = \begin{cases} 1 & \text{if } (r = r^* \wedge f(u^* \oplus c) = 1) \vee (f(\mathsf{PWPRF.PuncF}(\mathsf{fk}_{r^*}, r) \oplus c) = 1) \\ 0 & \text{otherwise} \end{cases}$$

Then, it returns the secret-key sk_f as the constrained key $\mathsf{fk}'_{C_{f,\mathsf{fk}_{r^*},u^*}} \leftarrow \mathsf{CWPRF.ConKey}(\mathsf{fk}', C_{f,\mathsf{fk}_{r^*},u^*})$. It updates the set $Q_f \leftarrow Q_f \cup \{f\}$.

– *decryption queries.* On input a ciphertext, function pair (ct, f), the challenger parses $\mathsf{ct} = (r, c, v)$. It defines a circuit $C_{f,\mathsf{fk}_{r^*},u^*}$ as above and computes $\mathsf{fk}'_{C_{f,\mathsf{fk}_{r^*},u^*}} \leftarrow \mathsf{CWPRF.ConKey}(\mathsf{fk}', C_{f,\mathsf{fk}_{r^*},u^*})$. Then, it sets $v' = \mathsf{CWPRF.ConF}(\mathsf{fk}'_{C_{f,\mathsf{fk}_{r^*},u^*}}, (r, c))$ and outputs 1 if $v' = v$, 0 otherwise. It updates $Q_{ct,f} \leftarrow Q_{ct,f} \cup (\mathsf{ct}, f)$.

Challenge: The adversary $\mathcal{A}$ submits a pair of challenge messages $(m_0, m_1) \in \{0,1\}^\ell \times \{0,1\}^\ell$. The challenger proceeds as follows:

1. Mask the challenge message as $c^* = m_b \oplus u^*$.
2. Compute a pseudorandom value $v^* = \mathsf{CWPRF.F}(\mathsf{fk}', (r^*, c^*))$.
3. Return the challenge ciphertext $\mathsf{ct}^* = (r^*, c^*, v^*)$ to $\mathcal{A}$.

Guess: $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. The challenger returns 1 if $b = b'$ and $f(m_0) = f(m_1)$ holds for all $f \in Q_f$ and for all $(\mathsf{ct}^*, f) \in Q_{ct,f}$.

We show in Lemma 1 that the advantage of $\mathcal{A}$ in distinguishing between the hybrids 2 and 3 is negligible in λ, however the proof is shifted to App. C.1.

Lemma 1 *Assuming wFP-IND security of CWPRF,* $|Pr[\mathsf{H}_2] - Pr[\mathsf{H}_3]| = negl(\lambda)$.

Hybd_4: This is exactly the same as Hybd_3 except that we pick u^* uniformly at random from the co-domain $\mathcal{Y}$ of $\mathsf{PWPRF.F}(\mathsf{fk}, \cdot)$ instead of computing $u^* \leftarrow \mathsf{PWPRF.F}(\mathsf{fk}, r^*)$ for some $r^* \in \{0,1\}^{2\lambda}$. The Sel-IND security of PWPRF guarantees that hybrids 3 and 4 are indistinguishable for any PPT adversary $\mathcal{A}$ as shown in Lemma 2. We prove this lemma in App. C.2.

Lemma 2 *Assuming Sel-IND security of PWPRF,* $|Pr[\mathsf{H}_3] - Pr[\mathsf{H}_4]| = negl(\lambda)$.

Finally, we note that in hybrid 4 the challenge message m_b is masked into $c^* = m_b \oplus u^*$ where u^* is chosen as uniformly at random from $\mathcal{Y}$, indicating that the challenge bit b is statistically hidden inside c^*. This in turn implies that $\mathcal{A}$'s advantage in guessing the bit b in hybrid 4 is at most $\frac{1}{2}$ even when it has access to the key generation and decryption oracles. This completes the proof of Adp-INDCCA security of our PKFE.

5 Conclusion

In this work, we propose a generalized variant of WPRF called constrained WPRF which provides finer access control to the pseudorandom values associated with NP statements. We discuss a generic construction of CWPRF from $i\mathcal{O}$ and CPRF. More importantly, the pseudorandomness and function privacy of our CWPRF enable us to achieve an adaptive IND-CCA secure PKFE for all polynomial-size functions. To the best of our knowledge, existing PKFEs are either IND-CPA secure or supports specific class of functions. Additionally, our PKFE produces optimal size ciphertexts which in turn implies succinctness. In literature, such a succinct PKFE gives rise to an $i\mathcal{O}$ for all circuits [4]. Thus, it can be believed that CWPRF is as good as $i\mathcal{O}$, however, a direct construction of $i\mathcal{O}$ from CWPRF would be more interesting which we leave as future work.

References

1. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In *Annual International Cryptology Conference*, pages 26–45. Springer, 1998.
2. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.
3. F. Benhamouda, F. Bourse, and H. Lipmaa. Cca-secure inner-product functional encryption from projective hash functions. In *IACR International Workshop on Public Key Cryptography*, pages 36–66. Springer, 2017.
4. N. Bitansky and V. Vaikuntanathan. Indistinguishability obfuscation from functional encryption. *Journal of the ACM (JACM)*, 65(6):1–37, 2018.
5. J. Blömer and G. Liske. Construction of fully cca-secure predicate encryptions from pair encoding schemes. In *Cryptographers' Track at the RSA Conference*, pages 431–447. Springer, 2016.
6. M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, page 103–112, New York, NY, USA, 1988. Association for Computing Machinery.
7. D. Boneh, S. Kim, and H. Montgomery. Private puncturable prfs from standard lattice assumptions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 415–445. Springer, 2017.
8. D. Boneh, K. Lewi, and D. J. Wu. Constraining pseudorandom functions privately. Cryptology ePrint Archive, Report 2015/1167, 2015. `https://eprint.iacr.org/2015/1167`.
9. D. Boneh, K. Lewi, and D. J. Wu. Constraining pseudorandom functions privately. In *IACR International Workshop on Public Key Cryptography*, pages 494–524. Springer, 2017.
10. D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography Conference*, pages 253–273. Springer, 2011.
11. D. Boneh and B. Waters. Constrained pseudorandom functions and their applications. In *International conference on the theory and application of cryptology and information security*, pages 280–300. Springer, 2013.
12. R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In *International conference on the theory and applications of cryptographic techniques*, pages 207–222. Springer, 2004.
13. D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. In *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*, STOC '91, page 542–552, New York, NY, USA, 1991. Association for Computing Machinery.
14. E. Fujisaki and T. Okamoto. How to enhance the security of public-key encryption at minimum cost. In *International Workshop on Public Key Cryptography*, pages 53–68. Springer, 1999.
15. S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM Journal on Computing*, 45(3):882–929, 2016.
16. S. Goldwasser, Y. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. Reusable garbled circuits and succinct functional encryption. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 555–564, 2013.

17. V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 89–98, 2006.
18. A. Jain, H. Lin, and A. Sahai. Indistinguishability obfuscation from well-founded assumptions. *arXiv preprint arXiv:2008.09317*, 2020.
19. E. Kiltz. Direct chosen-ciphertext secure identity-based encryption in the standard model with short ciphertexts, 2006.
20. F. Kitagawa and T. Matsuda. Cpa-to-cca transformation for kdm security. In *Theory of Cryptography Conference*, pages 118–148. Springer, 2019.
21. V. Koppula and B. Waters. Realizing chosen ciphertext security generically in attribute-based encryption and predicate encryption. In *Annual International Cryptology Conference*, pages 671–700. Springer, 2019.
22. T. Matsuda and G. Hanaoka. Chosen ciphertext security via point obfuscation. In *Theory of Cryptography Conference*, pages 95–120. Springer, 2014.
23. T. Matsuda and G. Hanaoka. Constructing and understanding chosen ciphertext security via puncturable key encapsulation mechanisms. In *Theory of Cryptography Conference*, pages 561–590. Springer, 2015.
24. M. Nandi and T. Pandit. Generic conversions from cpa to cca secure functional encryption. *IACR Cryptol. ePrint Arch.*, 2015:457, 2015.
25. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 427–437, 1990.
26. T. Pal and R. Dutta. Offline witness encryption from witness prf and randomized encoding in crs model. In *Australasian Conference on Information Security and Privacy*, pages 78–96. Springer, 2019.
27. T. Pal and R. Dutta. Chosen-ciphertext secure multi-identity and multi-attribute pure fhe. In *International Conference on Cryptology and Network Security*, pages 387–408. Springer, 2020.
28. T. Pal and R. Dutta. Semi-adaptively secure offline witness encryption from puncturable witness prf. In *International Conference on Provable Security*, pages 169–189. Springer, 2020.
29. C. Peikert and S. Shiehian. Noninteractive zero knowledge for np from (plain) learning with errors. In *Annual International Cryptology Conference*, pages 89–114. Springer, 2019.
30. C. Peikert and B. Waters. Lossy trapdoor functions and their applications. *SIAM Journal on Computing*, 40(6):1803–1844, 2011.
31. V. Shoup. *Why chosen ciphertext security matters*, volume 57. Citeseer, 1998.
32. S. Yamada, N. Attrapadung, G. Hanaoka, and N. Kunihiro. Generic constructions for chosen-ciphertext secure attribute based encryption. In *International Workshop on Public Key Cryptography*, pages 71–89. Springer, 2011.
33. T. Yamakawa, S. Yamada, G. Hanaoka, and N. Kunihiro. Adversary-dependent lossy trapdoor function from hardness of factoring semi-smooth rsa subgroup moduli. In *Annual International Cryptology Conference*, pages 3–32. Springer, 2016.
34. M. Zhandry. How to avoid obfuscation using witness prfs. In *Theory of Cryptography Conference*, pages 421–448. Springer, 2016.

A Pseudorandomness Security of PWPRF

In this section, we describe the pseudorandomness security notions of PWPRF = (Gen, PunKey, F, PunF, Eval) for an NP language L with a witness relation $R : \mathcal{X} \times \mathcal{W} \to \{0, 1\}$.

Definition 18 [Adaptive Pseudorandomness] For a security parameter $\lambda \in \mathbb{N}$, an NP language L with a relation $R : \mathcal{X} \times \mathcal{W} \to \{0,1\}$ and a bit $b \in \{0,1\}$, we define the experiment $\mathsf{Adp\text{-}IND}_{\mathcal{A}}^{\mathsf{PWPRF}}(1^\lambda, b)$ between a challenger and a PPT adversary $\mathcal{A}$ in the following manner:

Setup: The challenger runs $(\mathsf{fk}, \mathsf{ek}) \leftarrow \mathsf{Gen}(1^\lambda, R)$ and sends ek to $\mathcal{A}$. It also prepares an empty set Q_x.

Queries: After setup, $\mathcal{A}$ can make queries to the following oracle at any point of the experiment.

- *pseudorandom function queries.* On input $x \in \mathcal{X}$, the challenger returns a pseudorandom value $y \leftarrow \mathsf{F}(\mathsf{fk}, x)$ and updates the set $Q_x \leftarrow Q_x \cup \{x\}$.

Challenge: At some point, $\mathcal{A}$ submits a challenge string $x^* \in \mathcal{X} \setminus L$. The challenger computes $\mathsf{fk}_{x^*} \leftarrow \mathsf{PuncKey}(\mathsf{fk}, x^*)$ and sets $y_0 \leftarrow \mathsf{F}(\mathsf{fk}, x^*)$, $y_1 \leftarrow \mathcal{Y}$. Finally, it returns (fk_{x^*}, y_b) to $\mathcal{A}$.

Guess: Eventually, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. The challenger returns 1 if $b = b'$ and $x^* \notin Q_x$.

A PWPRF is said to be adaptively secure if the following advantage is a negligible function of λ:

$$\mathsf{Adv}^{\mathsf{Adp\text{-}IND}}_{\mathcal{A},\mathsf{PWPRF}}(\lambda) = |\Pr[\mathsf{Adp\text{-}IND}^{\mathsf{PWPRF}}_{\mathcal{A}}(1^\lambda, b) = 1] - \tfrac{1}{2}|$$

Definition 19 [Selective Pseudorandomness] We define a security experiment $\mathsf{Sel\text{-}IND}_{\mathcal{A}}^{\mathsf{PWPRF}}(1^\lambda, b)$ for selective security of PWPRF similarly to the experiment $\mathsf{Adp\text{-}IND}_{\mathcal{A}}^{\mathsf{PWPRF}}(1^\lambda, b)$ except that the adversary $\mathcal{A}$ submits the challenge statement $x^* \in \mathcal{X} \setminus L$ before setup phase. We define the advantage $\mathsf{Adv}^{\mathsf{Sel\text{-}IND}}_{\mathcal{A},\mathsf{PWPRF}}(\lambda)$ accordingly and say that the PWPRF is selectively secure if the quantity is a negligible function of λ.

B Security Analysis of CWPRF

B.1 Proof of Theorem 1

Proof. We prove the security using two hybrid experiments. The first one is Hybd_0 which is the standard experiment $\mathsf{Sel\text{-}IND}_{\mathcal{A}}^{\mathsf{CWPRF}}(1^\lambda, b)$ as described in Def. 11. In the second experiment Hybd_1, we modify the public evaluation key and argue the indistinguishability between the hybrids relying on the security of $i\mathcal{O}$. Finally, we conclude the proof using the Sel-IND security of CPRF.

Hybd_0: The challenger proceeds as specified in $\mathsf{Sel\text{-}IND}_{\mathcal{A}}^{\mathsf{CWPRF}}(1^\lambda, b)$ of Def. 11. Upon receiving the challenge statement $x^* \in \mathcal{X} \setminus L$ from the adversary $\mathcal{A}$, the challenger generates $\mathsf{msk} \leftarrow \mathsf{CPRF.Setup}(1^\lambda)$, computes a circuit $F_{\mathsf{msk},R}$ (as defined in the CWPRF construction above), obfuscates the circuit as $\widetilde{F} = i\mathcal{O}(1^\lambda, F_{\mathsf{msk},R})$, and sets $\mathsf{fk} = \mathsf{msk}$, $\mathsf{ek} = \widetilde{F}$. It computes $y_0 \leftarrow \mathsf{CPRF.Eval}(\mathsf{msk}, x^*)$ and $y_1 \leftarrow \mathcal{Y}$. The adversary $\mathcal{A}$ gets (ek, y_b) and makes query to the constrained key oracle $\mathsf{CWPRF.ConKey}(\mathsf{fk}, \cdot)$ and the pseudorandom function oracle $\mathsf{CWPRF.F}(\mathsf{fk}, \cdot)$ for any polynomial number of times. Let Q_c, Q_x be the set of all constrained key queries and pseudorandom function queries respectively, made

by $\mathcal{A}$ during the experiment. The challenger outputs 1 if $\mathcal{A}$ can guess the bit b correctly and $x^* \notin Q_c$, $C(x^*) = 0$ for all $C \in Q_c$.

Hybd_1: It is the same Hybd_0 except that the circuit $F_{\mathsf{msk},R}$ is modified to a new circuit $F_{\mathsf{sk}^*,R}$ given by

$$F_{\mathsf{sk}^*,R}(x,w) = \begin{cases} \mathsf{CPRF.ConEval}(\mathsf{sk}^*, x) & \text{if } R(x,w) = 1 \\ \perp & \text{otherwise} \end{cases}$$

where sk^* is defined as follows. The challenge statement x^* is used to build a circuit E_{x^*} such that $E_{x^*}(x) = 1$ for all $x \in \mathcal{X} \setminus \{x^*\}$ and $E_{x^*}(x^*) = 0$. Then the challenger computes a constrained key $\mathsf{sk}^* \leftarrow \mathsf{CPRF.ConKey}(\mathsf{msk}, E_{x^*})$ and generates $\mathsf{ek} = i\mathcal{O}(1^\lambda, F_{\mathsf{sk}^*,R})$. The rest of the experiment is the same as Hybd_0. First, we observe that the two circuits $F_{\mathsf{msk},R}$ and $F_{\mathsf{sk}^*,R}$ are equivalent. For any $x \in \mathcal{X} \setminus \{x^*\}$, it holds that $\mathsf{CPRF.Eval}(\mathsf{msk}, x) = \mathsf{CPRF.ConEval}(\mathsf{sk}^*, x)$ due to the functionality of E_{x^*}, which implies $F_{\mathsf{msk},R}(x,w) = F_{\mathsf{sk}^*,R}(x,w)$ for all $x \in \mathcal{X} \setminus \{x^*\}$ and $w \in \mathcal{W}$. By the descriptions of both the circuits, we see that $F_{\mathsf{msk},R}(x^*,w) = F_{\mathsf{sk}^*,R}(x^*,w) = \perp$ as $x^* \notin L$. Hence, for any PPT distinguisher, the distributions of $i\mathcal{O}(1^\lambda, F_{\mathsf{msk},R})$ and $i\mathcal{O}(1^\lambda, F_{\mathsf{sk}^*,R})$ are indistinguishable by the security of $i\mathcal{O}$ (Def. 17). Therefore, Hybd_0 and Hybd_1 are indistinguishable from $\mathcal{A}$'s point of view.

Next, we prove that the advantage of $\mathcal{A}$ in Hybd_1 is negligible based on the Sel-IND security of CPRF. Let $\mathcal{B}$ be an adversary of $\mathsf{Sel\text{-}IND}^{\mathsf{CPRF}}_{\mathcal{A}}(1^\lambda, b)$ experiment (Def. 5). Note that, $\mathcal{B}$ has access to the oracles $\mathsf{CPRF.ConKey}(\mathsf{msk}, \cdot)$ and $\mathsf{CPRF.Eval}(\mathsf{msk}, \cdot)$ where $\mathsf{msk} \leftarrow \mathsf{CPRF.Setup}(1^\lambda)$ is generated by the CPRF-challenger. The CWPRF-adversary $\mathcal{A}$ sends the challenge statement x^* and $\mathcal{B}$ simulates the experiment for $\mathcal{A}$ as follows:

$\underline{\mathcal{B}(1^\lambda, x^*)}$:

1. $\mathcal{B}$ sends x^* to its challenger and receives y_b according to the challenge bit b where $y_0 = \mathsf{CPRF.Eval}(\mathsf{msk}, x^*)$ and $y_1 \leftarrow \mathcal{Y}$.
2. Next, $\mathcal{B}$ computes E_{x^*} and asks for a constrained key from its challenger. It receives a key $\mathsf{sk}^* \leftarrow \mathsf{CPRF.ConKey}(\mathsf{msk}, E_{x^*})$.
3. $\mathcal{B}$ generates an obfuscated circuit $\mathsf{ek} = i\mathcal{O}(1^\lambda, F_{\mathsf{sk}^*,R})$ for a circuit $F_{\mathsf{sk}^*,R}$ as described in Hybd_1 and sends (ek, y_b) to $\mathcal{A}$.
4. Whenever $\mathcal{B}$ receives a constrained key query from $\mathcal{A}$ corresponding to a circuit C satisfying $C(x^*) = 0$, it uses the oracle $\mathsf{CPRF.ConKey}(\mathsf{msk}, \cdot)$ to produce a reply for $\mathcal{A}$.
5. $\mathcal{A}$ makes pseudorandom function query for a value $x \in \mathcal{X} \setminus \{x^*\}$ to which $\mathcal{B}$ replies with the oracle $\mathsf{CPRF.Eval}(\mathsf{msk}, \cdot)$.
6. Finally, $\mathcal{A}$ submits a guess bit b' which $\mathcal{B}$ forwards to its challenger.

First we note that $\mathcal{B}$ is an admissible adversary since $C(x^*) = 0$ for all $C \in Q_c \cup \{E_{x^*}\}$ where Q_c is the set of constrained key queries made by $\mathcal{A}$ and $x \neq x^*$ for all $x \in Q_x$ where Q_x denotes the set of pseudorandom function queries of $\mathcal{A}$. Hence, the advantage of $\mathcal{B}$ in breaking the Sel-IND security of CPRF is the same as the advantage of $\mathcal{A}$ in guessing the challenge bit in Hybd_1. The advantage of

Constants: a constrained key sk^*, a string x^*, a value r^* and the relation circuit R
Inputs: a statement $x \in \mathcal{X}$ and a witness $w \in \mathcal{W}$

1. If $R(x, w) = 1$
2. If $x \neq x^*$
3. return $\mathsf{CPRF.ConEval}(\mathsf{sk}^*, x)$
4. If $x = x^*$
5. return r^*
6. else return $\perp$

Fig. 1: Description of circuit $F_{\mathsf{sk}^*, x^*, r^*, R}$

$\mathcal{A}$ in breaking the Sel-IND security of CWPRF is negligible in λ since the CPRF is Sel-IND secure.

B.2 Proof of Theorem 2

Proof. The security analysis involves two hybrid experiments where the first hybrid is the standard experiment $\mathsf{wFP\text{-}IND}_{\mathcal{A}}^{\mathsf{CWPRF}}(1^\lambda, b)$ between an adversary $\mathcal{A}$ and a challenger as described in Def. 13 and the second hybrid modifies the public evaluation key depending on the security of the $i\mathcal{O}$. Finally, we show that the advantage of $\mathcal{A}$ in the second hybrid is negligible due to the wFP-IND security of the CPRF.

Hybd_0: It is the same as $\mathsf{wFP\text{-}IND}_{\mathcal{A}}^{\mathsf{CWPRF}}(1^\lambda, b)$ of Def. 13. The challenger generates $\mathsf{msk} \leftarrow \mathsf{CPRF.Setup}(1^\lambda)$, sets the secret function key $\mathsf{fk} = \mathsf{msk}$ and sends the public evaluation key $\mathsf{ek} = i\mathcal{O}(1^\lambda, F_{\mathsf{msk},R})$ to $\mathcal{A}$ where the circuit $F_{\mathsf{msk},R}$ is as defined in the construction. Upon receiving ek, the adversary makes two kind of queries. It sends a pair of circuits (C_0, C_1) for a constrained key query to which $\mathcal{A}$ receives a key of the form $\mathsf{CWPRF.ConKey}(\mathsf{fk}, C_b)$ and it can ask pseudorandom value for an input $x \in \mathcal{X}$ to which $\mathcal{A}$ receives $\mathsf{CWPRF.F}(\mathsf{fk}, x)$. Let $Q_{c,c}$ be the set of all constrained key queries made by $\mathcal{A}$ during the experiment. The challenger outputs 1 if $\mathcal{A}$ can guess the bit b correctly and $C_0(x) = C_1(x)$ for all $(C_0, C_1) \in Q_{c,c}$ and $x \in \mathcal{X}$.

Hybd_1: It is the same Hybd_0 except that the circuit $F_{\mathsf{msk},R}$ is replaced with a new circuit $F_{\mathsf{sk}^*, x^*, r^*, R}$ described in Fig. 1. The challenger picks an arbitrary $x^* \in \mathcal{X}$ and define a circuit E_{x^*} such that $E_{x^*}(x) = 1$ for all $x \in \mathcal{X} \setminus \{x^*\}$ and $E_{x^*}(x^*) = 0$. Then it computes a constrained key $\mathsf{sk}^* \leftarrow \mathsf{CPRF.ConKey}(\mathsf{msk}, E_{x^*})$ and generates $\mathsf{ek} = i\mathcal{O}(1^\lambda, F_{\mathsf{sk}^*, x^*, r^*, R})$ where $r^* \leftarrow \mathsf{CPRF.Eval}(\mathsf{msk}, x^*)$. The rest of the experiment is the same as the standard one described in Hybd_0. It is trivial to observe that the two circuits $F_{\mathsf{msk},R}$ and $F_{\mathsf{sk}^*, x^*, r^*, R}$ are functionally equivalent since $\mathsf{CPRF.Eval}(\mathsf{msk}, x) = \mathsf{CPRF.ConEval}(\mathsf{sk}^*, x)$ for any $x \in \mathcal{X} \setminus \{x^*\}$ due to the definition of E_{x^*}. Therefore, for any PPT distinguisher, the distributions of $i\mathcal{O}(1^\lambda, F_{\mathsf{msk},R})$ and $i\mathcal{O}(1^\lambda, F_{\mathsf{sk}^*, x^*, r^*, R})$ are indistinguishable by the security of $i\mathcal{O}$ (Def. 17). Therefore, the advantage of $\mathcal{A}$ in distinguishing between Hybd_0 and Hybd_1 is negligible.

Next, we conclude the proof by showing that the advantage of $\mathcal{A}$ in guessing the challenge bit in Hybd_1 is negligible based on the wFP-IND security of

CPRF. Let $\mathcal{B}$ be an adversary of wFP-IND experiment (Def. 7) of CPRF. Note that, $\mathcal{B}$ has access of two oracles: the first one returns $\mathsf{CPRF.ConKey}(\mathsf{msk}, C_b)$ on input (C_0, C_1) according to the challenge bit b and the second one outputs $\mathsf{CPRF.Eval}(\mathsf{msk}, x)$ on input a string $x \in \mathcal{X}$ where $\mathsf{msk} \leftarrow \mathsf{CPRF.Setup}(1^\lambda)$ is generated by the CPRF-challenger. The adversary $\mathcal{B}$ simulates the experiment for $\mathcal{A}$ as follows:

$\underline{\mathcal{B}(1^\lambda)}$:

1. $\mathcal{B}$ selects a random x^* from $\mathcal{X}$ and defines E_{x^*} as in the previous hybrid. It asks a constrained key query for (E_{x^*}, E_{x^*}) and an evaluation query for x^* from its challenger. In response, it receives a key sk^* and a pseudorandom value r^*.
2. $\mathcal{B}$ generates an obfuscated circuit $\mathsf{ek} = i\mathcal{O}(1^\lambda, F_{\mathsf{sk}^*, x^*, r^*, R})$ for a circuit $F_{\mathsf{sk}^*, x^*, r^*, R}$ as described in Fig. 1 and sends ek to $\mathcal{A}$.
3. Whenever $\mathcal{B}$ receives a constrained key query for a pair of equivalent circuits (C_0, C_1) from $\mathcal{A}$, it forwards to its challenger who returns a key $\mathsf{CPRF.ConKey}(\mathsf{msk}, C_b)$. Then, $\mathcal{B}$ sends the key to $\mathcal{A}$.
4. $\mathcal{A}$ makes pseudorandom function query for a value $x \in \mathcal{X}$ to which $\mathcal{B}$ replies with the oracle $\mathsf{CPRF.Eval}(\mathsf{msk}, \cdot)$.
5. Finally, $\mathcal{A}$ submits a guess bit b' which $\mathcal{B}$ forwards to its challenger.

We observe that $\mathcal{B}$ is an admissible adversary since for all $(C_0, C_1) \in Q_{c,c} \cup \{(E_{x^*}, E_{x^*})\}$ it holds that $C_0(x) = C_1(x)\ \forall x \in \mathcal{X}$. Therefore, $\mathcal{B}$ perfectly simulates Hybd_1 for $\mathcal{A}$. This implies that the advantage of $\mathcal{A}$ in guessing the challenge bit is negligible in λ by wFP-IND security of the CPRF.

C Missing Parts of Theorem 3

C.1 Proof of Lemma 1

Proof. We will prove this by contradiction. Suppose, the PKFE adversary $\mathcal{A}$'s advantage in hybrids 2 differs by a non-negligible quantity from its advantage in hybrid 3, i.e. there exists a polynomial $p(\lambda)$ such that

$$|\Pr[\mathsf{H}_2] - \Pr[\mathsf{H}_3]| \geq \tfrac{1}{p(\lambda)}$$

holds for sufficiently many $\lambda \in \mathbb{N}$. We use $\mathcal{A}$ to construct a CWPRF adversary $\mathcal{B}$ for the wFP-IND security experiment $\mathsf{wFP\text{-}IND}_{\mathcal{B}}^{\mathsf{CWPRF}}(1^\lambda, \beta)$ as described in Def. 13 for some $\beta \in \{0, 1\}$. For a key pair $(\mathsf{fk}, \mathsf{ek}) \leftarrow \mathsf{PWPRF.Gen}(1^\lambda, R)$, the CWPRF-challenger generates $(\mathsf{fk}', \mathsf{ek}') \leftarrow \mathsf{CWPRF.Gen}(1^\lambda, R_{\mathsf{ek}})$ and sends ek' to $\mathcal{B}$. We note that the NP relation circuit R_{ek} is public and the PWPRF key pair (fk, ek) is made available to $\mathcal{B}$ by the CWPRF challenger as an auxiliary information. There are two oracles to which $\mathcal{B}$ can query. Firstly, it can send a pair of equivalent circuits (C_0, C_1) and learn a constrained key $\mathsf{fk}'_{C_\beta} \leftarrow \mathsf{CWPRF.ConKey}(\mathsf{fk}', C_\beta)$. Secondly, it may send a string x and learn a pseudorandom value $y \leftarrow \mathsf{CWPRF.F}(\mathsf{fk}', x)$. Now, $\mathcal{B}$ simulates the adversary $\mathcal{A}$ as follows.

$\underline{\mathcal{B}(1^\lambda, (\mathsf{fk}, \mathsf{ek}), \mathsf{ek}')}$:

1. It sends the master public-key $\mathsf{pp} = (\mathsf{ek}, \mathsf{ek}')$ to $\mathcal{A}$.
2. It picks a random string $r^* \leftarrow \{0,1\}^{2\lambda}$ and computes the punctured key $\mathsf{fk}_{r^*} \leftarrow \mathsf{PWPRF.PuncKey}(\mathsf{fk}, r^*)$ and the pseudorandom value $u^* \leftarrow \mathsf{PWPRF.F}(\mathsf{fk}, r^*)$. It takes an empty set Q_f.
3. Whenever $\mathcal{A}$ queries for a secret-key corresponding to a function f, the adversary $\mathcal{B}$ computes a pair of circuits as $\widehat{C}_{f,0} = C_{f,\mathsf{fk}}$ (as defined in the PKFE construction) and $\widehat{C}_{f,1} = C_{f,\mathsf{fk}_{r^*},u^*}$. Now, $\mathcal{B}$ sends $(\widehat{C}_{f,0}, \widehat{C}_{f,1})$ to the constrained key oracle and receives a constrained key $\mathsf{fk}'_{\widehat{C}_{f,\beta}} \leftarrow \mathsf{CWPRF.ConKey}(\mathsf{fk}', \widehat{C}_{f,\beta})$ which is forwarded to $\mathcal{A}$ as the secret-key sk_f. The adversary $\mathcal{B}$ updates the set $Q_f \leftarrow Q_f \cup \{f\}$.
4. $\mathcal{A}$ can also query for decryption of a ciphertext-function pair $(\mathsf{ct} = (r, c, v), f)$ at any point of the experiment. First, $\mathcal{B}$ creates the circuit pair $(\widehat{C}_{f,0}, \widehat{C}_{f,1})$ as in step 3 and learns a constrained key $\mathsf{fk}'_{\widehat{C}_{f,\beta}} \leftarrow \mathsf{CWPRF.ConKey}(\mathsf{fk}', \widehat{C}_{f,\beta})$ from its oracle. Then, it returns 1 if $v = \mathsf{CWPRF.ConF}(\mathsf{fk}'_{\widehat{C}_{f,\beta}}, (r, c))$, 0 otherwise. The adversary $\mathcal{B}$ updates the sets $Q_f \leftarrow Q_f \cup \{f\}$.
5. Let $m_0, m_1 \in \{0,1\}^\ell$ be the challenge messages given by $\mathcal{A}$. Now, $\mathcal{B}$ chooses $b \leftarrow \{0,1\}$ and proceeds by masking the challenge message as $c^* = m_b \oplus u^*$. Then, $\mathcal{B}$ makes a pseudorandom function query on the input (r^*, c^*) and gets $v^* = \mathsf{CWPRF.F}(\mathsf{fk}', (r^*, c^*))$. Finally, it returns the challenge ciphertext as $\mathsf{ct}^* = (r^*, c^*, v^*)$.
6. At the end, $\mathcal{A}$ outputs a guess bit which $\mathcal{B}$ returns as its own guess.

First, we show that $\mathcal{B}$ is an admissible adversary of the wFP-IND game. The correctness of PWPRF.PuncF, it holds that $\mathsf{PWPRF.F}(\mathsf{fk}, r)$ (used in the circuits $C_{f,\mathsf{fk}}$) is equal to $\mathsf{PWPRF.PuncF}(\mathsf{fk}_{r^*}, r)$ (used in the circuits $C_{f,\mathsf{fk}_{r^*},u^*}$) for all $r \neq r^*$. Therefore, the circuits $C_{f,\mathsf{fk}}$ and $C_{f,\mathsf{fk}_{r^*},u^*}$ are equivalent for all $f \in Q_f$. Now, if $\beta = 0$, then $\mathcal{B}$ perfectly simulates the hybrid 2 as the secret-keys are computed using the circuits of the form $C_{f,\mathsf{fk}}$ where the secret function key fk is hardcoded. If $\beta = 1$ then $\mathcal{B}$ perfectly simulates the hybrid 3 as the secret-keys are produced utilizing the circuits of the form $C_{f,\mathsf{fk}_{r^*},u^*}$ where the punctured key fk_{r^*} is constant. Therefore, for infinitely many λ, it holds that

$$\mathsf{Adv}^{\mathsf{wFP\text{-}IND}}_{\mathcal{B},\mathsf{CWPRF}}(\lambda) = |\Pr[\mathsf{H}_2] - \Pr[\mathsf{H}_3]| \geq \tfrac{1}{p(\lambda)}.$$

This is a contradiction as CWPRF is wFP-IND secure and hence we have $|\Pr[\mathsf{H}_2] - \Pr[\mathsf{H}_3]| = negl(\lambda)$.

C.2 Proof of Lemma 2

Proof. We prove the lemma by contradiction. We assume that PKFE adversary $\mathcal{A}$'s advantage in hybrids 3 differs by a non-negligible quantity from its advantage in hybrid 4. Consequently, there exists a polynomial $p(\lambda)$ such that

$$|\Pr[\mathsf{H}_3] - \Pr[\mathsf{H}_4]| \geq \tfrac{1}{p(\lambda)}$$

holds for sufficiently many $\lambda \in \mathbb{N}$. Now, we construct a PWPRF adversary $\mathcal{B}$ for the $\mathsf{Sel\text{-}IND}$ security experiment $\mathsf{Sel\text{-}IND}^{\mathsf{PWPRF}}_{\mathcal{B}}(1^\lambda, \beta)$ as described in Def. 19 for some $\beta \in \{0,1\}$. First, $\mathcal{B}$ chooses a challenge statement $r^* \leftarrow \{0,1\}^{2\lambda}$ and sends it to the PWPRF-challenger. The challenger picks a challenge bit β and generates a key pair $(\mathsf{fk}, \mathsf{ek}) \leftarrow \mathsf{PWPRF.Gen}(1^\lambda, R)$. It computes a punctured key $\mathsf{fk}_{r^*} \leftarrow \mathsf{PWPRF.PuncKey}(\mathsf{fk}, r^*)$ and a pseudorandom value u^* depending on β. Finally, $\mathcal{B}$ receives $(\mathsf{ek}, \mathsf{fk}_{r^*}, u^*)$ from it's challenger. Now, $\mathcal{B}$ simulates the adversary $\mathcal{A}$ as follows.

$\underline{\mathcal{B}(1^\lambda, \mathsf{ek}, \mathsf{fk}_{r^*}, r^*, u^*)}$:

1. It generates a key pair $(\mathsf{fk}', \mathsf{ek}') \leftarrow \mathsf{CWPRF.Gen}(1^\lambda, R_{\mathsf{ek}})$ and sends the master public-key $\mathsf{pp} = (\mathsf{ek}, \mathsf{ek}')$ to $\mathcal{A}$.
2. To answer secret-key query of $\mathcal{A}$ corresponding to a function f, the adversary $\mathcal{B}$ computes a circuit as $C_{f,\mathsf{fk}_{r^*},u^*}$ (as defined in hybrid 3 of Theorem 3) and sends the secret-key sk_f as the constrained key $\mathsf{fk}'_{C_{f,\mathsf{fk}_{r^*},u^*}} \leftarrow \mathsf{CWPRF.ConKey}(\mathsf{fk}', C_{f,\mathsf{fk}_{r^*},u^*})$.
3. $\mathcal{A}$ can also query for decryption of a ciphertext-function pair $(\mathsf{ct} = (r, c, v), f)$ at any point of the experiment. First, $\mathcal{B}$ computes a constrained key $\mathsf{fk}'_{C_{f,\mathsf{fk}_{r^*},u^*}} \leftarrow \mathsf{CWPRF.ConKey}(\mathsf{fk}', C_{f,\mathsf{fk}_{r^*},u^*})$ and then, it returns 1 if $v = \mathsf{CWPRF.ConF}(\mathsf{fk}'_{C_{f,\mathsf{fk}_{r^*},u^*}}, (r, c))$, 0 otherwise.
4. At a certain stage, $\mathcal{A}$ submits challenge messages $m_0, m_1 \in \{0,1\}^\ell$. Now, $\mathcal{B}$ chooses $b \leftarrow \{0,1\}$ and proceeds by masking the challenge message as $c^* = m_b \oplus u^*$ by the pseudorandom value u^*. Then, it computes a pseudorandom value $v^* = \mathsf{CWPRF.F}(\mathsf{fk}', (r^*, c^*))$ and returns the challenge ciphertext as $\mathsf{ct}^* = (r^*, c^*, v^*)$.
5. Finally, $\mathcal{A}$ outputs a guess bit which $\mathcal{B}$ returns as its own guess.

Note that $\mathcal{B}$ is an admissible adversary for the $\mathsf{Sel\text{-}IND}$ security experiment if r^* does not have any witness under the relation R. Since PRG is a length doubling pseudorandom generator with input length λ and r^* is choosen uniformly at random from $\{0,1\}^{2\lambda}$, the probability that there exists $s \in \{0,1\}^\lambda$ satisfying $\mathsf{PRG}(s) = r^*$ is at most $2^{-\lambda}$ which is negligible in λ. Therefore, with overwhelming probability, $\mathcal{B}$ is a legitimate adversary for the $\mathsf{Sel\text{-}IND}$ security game of PWPRF. If the challenge bit β chosen by the PWPRF-challenger is 0, then the pseudorandom value u^* is computed as $u^* = \mathsf{PWPRF.F}(\mathsf{fk}, r^*)$ and when $\beta = 1$ then u^* is chosen uniformly at random from the co-domain $\mathcal{Y}$ of $\mathsf{PWPRF.F}(\mathsf{fk}, \cdot)$. Thus, $\mathcal{B}$ simulates hybrid 3 if $\beta = 0$ and it simulates hybrid 4 if $\beta = 1$. Hence, for infinitely many λ, it holds that

$$\mathsf{Adv}^{\mathsf{Sel\text{-}IND}}_{\mathcal{B},\mathsf{PWPRF}}(\lambda) = |\Pr[\mathsf{H}_3] - \Pr[\mathsf{H}_4]| \geq \tfrac{1}{p(\lambda)}$$

which contradicts the fact that PWPRF is $\mathsf{Sel\text{-}IND}$ secure. Therefore, we must have $|\Pr[\mathsf{H}_2] - \Pr[\mathsf{H}_3]| = negl(\lambda)$.

D Application of CWPRF in Tag based ABE

D.1 Attribute Based Encryption

Definition 20 [Attribute Based Encryption with Tag] An attribute based encryption (ABE) with tag is defined for a class of functions $\{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$, a class of attributes $\{\Im_\lambda\}_{\lambda\in\mathbb{N}}$, a tag space $\mathcal{T}_\lambda$ and a message space $\mathcal{M}$. It consists of four PPT algorithms Setup, KeyGen, Enc, Dec that work as follows:

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{msk}, \mathsf{pp})$: The Setup algorithm takes as input a security paramenter λ and outputs a master secret-key msk and a public parameter pp.
- $\mathsf{KeyGen}(\mathsf{msk}, f, \tau) \to \mathsf{sk}_f$: The key generation algorithm takes as input the master secret-key msk, a circuit $f \in \mathcal{F}_\lambda$ and a tag $\tau \in \mathcal{T}_\lambda$ and outputs a secret-key sk_f.
- $\mathsf{Enc}(\mathsf{pp}, a, m, \tau') \to \mathsf{ct}$: The encryption algorithm takes as input the public parameter pp, an attribute $a \in \Im_\lambda$, a message $m \in \mathcal{M}$ and a tag $\tau' \in \mathcal{T}_\lambda$, and outputs a ciphertext ct. We assume that a and τ' is trivially included in the ciphertext.
- $\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}) \to y$: The decryption algorithm takes as input a secret-key sk_f and a ciphertext ct and outputs a value y.

An ABE with tag must satisfy the following requirements.

Definition 21 (Correctness) For all $\lambda \in \mathbb{N}, (\mathsf{msk}, \mathsf{pp}) \leftarrow \mathsf{Setup}(1^\lambda), a \in \Im_\lambda, \tau \in \mathcal{T}_\lambda, m \in \mathcal{M}, f \in \mathcal{F}_\lambda$ and $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f, \tau)$, we have

$$\Pr[\mathsf{Dec}(\mathsf{sk}_f, \mathsf{Enc}(\mathsf{pp}, a, m, \tau)) = m \text{ s.t. } f(m) = 1] = 1 - negl(\lambda)$$

Definition 22 [Adaptive Indistinguishability CCA security] For a security parameter $\lambda \in \mathbb{N}$, a function class $\{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$, an attribute class $\{\Im_\lambda\}_{\lambda\in\mathbb{N}}$, a message space $\mathcal{M}$, a tag space $\mathcal{T}_\lambda$ and a bit $b \in \{0,1\}$, we define the experiment $\mathsf{Adp\text{-}INDCCA}^{\mathsf{ABE}}_{\mathcal{A}}(1^\lambda, b)$ between a challenger and a PPT adversary $\mathcal{A}$ in the following manner:

Setup: The challenger runs $(\mathsf{msk}, \mathsf{pp}) \leftarrow \mathsf{Setup}(1^\lambda)$ and sends pp to $\mathcal{A}$. It also prepares two empty sets Q_f and Q_τ.

Queries: After setup, $\mathcal{A}$ can query to the following oracles at any point of the experiment.

- *secret-key queries.* On input a function $f \in \mathcal{F}_\lambda$ and a tag $\tau \in \mathcal{T}_\lambda$, the challenger returns a secret-key $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f, \tau)$ and updates the set $Q_f \leftarrow Q_f \cup \{f\}$.
- *decryption queries.* On input a ciphertext-function-tag tuple (ct, f, τ), the challenger computes $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f, \tau)$ and returns $\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct})$. It also updates $Q_\tau \leftarrow Q_\tau \cup \{\tau\}$.

Challenge: At some point, $\mathcal{A}$ submits a challenge attribute $a^* \in \Im_\lambda$, a challenge tag $\tau^* \in \mathcal{T}_\lambda$ and a pair of challenge messages $(m_0, m_1) \in \mathcal{M} \times \mathcal{M}$. The challenger returns $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pp}, a^*, m_b, \tau^*)$ to $\mathcal{A}$.

Guess: Eventually, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. The challenger returns 1 if $b = b'$ and $f(a^*) = 0$ holds for all $f \in Q_f$ and $\tau^* \notin Q_\tau$.

An ABE with tag is said to be adaptive indistinguishability CCA secure if the following advantage is a negligible function of λ:

$$\mathsf{Adv}^{\mathsf{Adp\text{-}INDCCA}}_{\mathcal{A},\mathsf{ABE}}(\lambda) = |\Pr[\mathsf{Adp\text{-}INDCCA}^{\mathsf{ABE}}_{\mathcal{A}}(1^\lambda, b) = 1] - \tfrac{1}{2}|$$

Definition 23 [Selective Indistinguishability CCA security] We define a security experiment $\mathsf{Sel\text{-}INDCCA}^{\mathsf{ABE}}_{\mathcal{A}}(1^\lambda, b)$ for selective security of ABE with tag similarly to the experiment $\mathsf{Adp\text{-}INDCCA}^{\mathsf{ABE}}_{\mathcal{A}}(1^\lambda, b)$ except that the adversary $\mathcal{A}$ submits the challenge attribute $a^* \in \Im_\lambda$ and the challenge tag $\tau^* \in \mathcal{T}_\lambda$ before setup phase. We define the advantage $\mathsf{Adv}^{\mathsf{Sel\text{-}INDCCA}}_{\mathcal{A},\mathsf{ABE}}(\lambda)$ accordingly and say that the ABE is selectively secure if the quantity is a negligible function of λ.

D.2 Construction of CCA secure ABE from CWPRF

In this section, we describe our construction of a CCA secure ABE *with tag* for all circuits from CWPRF. We require Sel-IND security of CWPRF to establish the security of the ABE. Our construction is inspired by the CCA secure ABE of Pal et al. [27] where they used a plain WPRF, a non-interactive proof system and a commitment scheme. However, we note that our ABE is solely based on CWPRF and achieves optimal size ciphertext similar to [27]. In particular, a ciphertext for message m is of size $|m| + 2\lambda$ where λ is the security parameter and $|m|$ denotes the size of m.

Construction. We build an $\mathsf{ABE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for a function class $\{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$, an attribute space $\{0,1\}^\lambda$, a tag space $\{0,1\}^t$ and a message space $\{0,1\}^\ell$. We consider a length doubling PRG with domain $\{0,1\}^\lambda$ for some $\lambda \in \mathbb{N}$ and a $\mathsf{CWPRF} = (\mathsf{Gen}, \mathsf{ConKey}, \mathsf{F}, \mathsf{ConF}, \mathsf{Eval})$ for the NP language $L = \{(a, r, \tau) \in \{0,1\}^\lambda \times \{0,1\}^{2\lambda} \times \{0,1\}^t : \exists\ s \text{ s.t. } \mathsf{PRG}(a \oplus s) = r\}$ with relation R. We assume that the CWPRF is associated with a class of boolean functions taking inputs from $\{0,1\}^{3\lambda+t}$.

$\mathsf{ABE.Setup}(1^\lambda)$: The setup generates $(\mathsf{fk}, \mathsf{ek}) \leftarrow \mathsf{CWPRF.Gen}(1^\lambda, R)$ and returns the master secret-key as $\mathsf{msk} = \mathsf{fk}$ and the public parameter as $\mathsf{pp} = \mathsf{ek}$.

$\mathsf{ABE.KeyGen}(\mathsf{msk}, f, \tau)$: On input a function $f \in \mathcal{F}_\lambda$ and a tag $\tau \in \{0,1\}^t$, the key generation algorithm builds a circuit $C_{f,\tau}$ such that

$$C_{f,\tau}(a, r, \tau') = \begin{cases} 1 & \text{if } \tau = \tau' \wedge f(a) = 1 \\ 0 & \text{otherwise} \end{cases}$$

Next, it computes the constrained key $\mathsf{fk}_{C_{f,\tau}} \leftarrow \mathsf{CWPRF.ConKey}(\mathsf{fk}, C_{f,\tau})$ and returns the secret-key $\mathsf{sk}_f = \mathsf{fk}_{C_{f,\tau}}$.

$\mathsf{ABE.Enc}(\mathsf{pp}, a, m, \tau')$: The encryption algorithm selects the randomness $s \leftarrow \{0,1\}^\lambda$ and encodes the attribute $a \in \{0,1\}^\lambda$ by computing $r = \mathsf{PRG}(a \oplus s) \in \{0,1\}^{2\lambda}$. Next, it masks the message $m \in \{0,1\}^\ell$ with respect to the tag τ' as $c = m \oplus \mathsf{CWPRF.Eval}(\mathsf{ek}, (a, r, \tau'), s)$. Finally, it returns the ciphertext $\mathsf{ct} = (a, r, c)$ along with the tag τ'.

$\mathsf{ABE.Dec}(\mathsf{sk}_f, \mathsf{ct})$: The decryption proceeds by parsing the secret-key $\mathsf{sk}_f = \mathsf{fk}_{C_{f,\tau}}$ corresponding to the tag τ and the ciphertext $\mathsf{ct} = (a, r, c)$ associated with a tag τ' and returns the message as $c \oplus \mathsf{CWPRF.ConF}(\mathsf{fk}_{C_{f,\tau}}, (a, r, \tau'))$.

Correctness. To demonstrate the correctness of our ABE, we take a secret-key $\mathsf{sk}_f \leftarrow \mathsf{ABE.KeyGen}(\mathsf{msk}, f, \tau)$ corresponding to a function f and a ciphertext $\mathsf{ct} \leftarrow \mathsf{ABE.Enc}(\mathsf{pp}, a, m, \tau)$ encrypting a message m with respect to an attribute a and tag τ using a randomness s such that $f(a) = 1$. Noting that $\mathsf{sk}_f = \mathsf{fk}_{C_{f,\tau}}$, the correctness of CWPRF guarantees that

$$\begin{aligned} \mathsf{CWPRF.F}(\mathsf{fk}, (a, r, \tau)) &= \mathsf{CWPRF.ConF}(\mathsf{fk}_{C_{f,\tau}}, (a, r, \tau)) && \text{as } C_{f,\tau}(a, r, \tau) = 1 \\ &= \mathsf{CWPRF.Eval}(\mathsf{ek}, (a, r, \tau), s) && \text{as } R((a, r, \tau), s) = 1 \\ &= c \oplus m && \text{as } \mathsf{ct} \text{ is well-formed} \end{aligned}$$

This ensures that $c \oplus \mathsf{CWPRF.ConF}(\mathsf{fk}_{C_{f,\tau}}, (a, r, \tau)) = m$ and hence the decryption is able to recover the message m using the secret-key sk_f as required.

Theorem 4 *The attribute-based encryption scheme ABE with tag described above is Adp-INDCCA secure as per Def. 22 (resp. Sel-INDCCA secure as par Def. 23) assuming the PRG is a secure as per Def. 1 and the CWPRF is Adp-IND secure as per Def. 10 (resp. Sel-IND secure as per Def. 11).*

Proof. We prove the adaptive security of the ABE as the selective counterpart will follow similarly. The proof proceeds in a sequence of hybrid experiments where the first experiment is identical to the Adp-INDCCA experiment of ABE as described in Def. 22 and the second experiment uses the master secret-key to encrypt the challenge message. In the third hybrid, the challenger encrypts the message using a random statement which does not have any valid witness under the relation R. Finally, we argue that the advantage of the adversary in guessing the challenge bit is negligible in the third hybrid using the pseudorandomness of CWPRF. Let us denote H_i by the event where the challenger outputs 1 in the i-th hybrid experiment.

Hybd$_0$: For any PPT adversary $\mathcal{A}$, we describe hybrid 0 which is the standard $\mathsf{Adp\text{-}INDCCA}(1^\lambda, b)$ experiment as described in Def. 22

Setup: The challenger generates a pair of keys $(\mathsf{fk}, \mathsf{ek}) \leftarrow \mathsf{CWPRF.Gen}(1^\lambda, R)$ and sends $\mathsf{pp} = \mathsf{ek}$ to $\mathcal{A}$. It creates two empty sets Q_f and Q_τ.
Queries: Now, $\mathcal{A}$ can query to the following oracles at any point of the experiment.

- *secret-key queries.* On input a function $f \in \mathcal{F}_\lambda$ and a tag τ, the challenger defines a circuit $C_{f,\tau}$ (as in the construction of ABE) and returns the secret-key sk_f as the constrained key $\mathsf{fk}_{C_{f,\tau}} \leftarrow \mathsf{CWPRF.ConKey}(\mathsf{fk}, C_{f,\tau})$. It updates the set $Q_f \leftarrow Q_f \cup \{f\}$.
- *decryption queries.* On input a ciphertext-function-tag tuple (ct, f, τ), the challenger parses $\mathsf{ct} = (a, r, c)$. It computes $\mathsf{fk}_{C_{f,\tau}} \leftarrow \mathsf{CWPRF.ConKey}(\mathsf{fk}, C_{f,\tau})$ and returns $c \oplus \mathsf{CWPRF.ConF}(\mathsf{fk}_{C_{f,\tau}}, (a, r, \tau))$. It updates the set $Q_\tau \leftarrow Q_\tau \cup \{\tau\}$.

Challenge: The adversary $\mathcal{A}$ submits a challenge attribute $a^* \in \{0,1\}^\lambda$, a challenge tag $\tau^* \in \{0,1\}^t$ and a pair of challenge messages $(m_0, m_1) \in \{0,1\}^\ell \times \{0,1\}^\ell$. The challenger proceeds as follows:

1. Pick $s^* \leftarrow \{0,1\}^\lambda$ and set $r^* = \mathsf{PRG}(a^* \oplus s^*)$.
2. Mask the challenge message as $c^* = m_b \oplus \mathsf{CWPRF.Eval}(\mathsf{ek}, (a^*, r^*, \tau^*), s^*)$.
3. Return the challenge ciphertext $\mathsf{ct}^* = (a^*, r^*, c^*)$ to $\mathcal{A}$.

Guess: $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. The challenger returns 1 if $b = b'$ and $f(a^*) = 0$ holds for all $f \in Q_f$ and $\tau^* \notin Q_\tau$.

Hybd_1: It is exactly the same as hybrid 0 except that the challenger makes use of the master secret-key $\mathsf{msk} = \mathsf{fk}$ instead of ek to evaluate the challenge ciphertext ct^* as shown below:

Challenge: The adversary $\mathcal{A}$ submits a challenge attribute $a^* \in \{0,1\}^\lambda$, a challenge tag $\tau^* \in 0,1^t$ and a pair of challenge messages $(m_0, m_1) \in \{0,1\}^\ell \times \{0,1\}^\ell$. The challenger proceeds as follows:

1. Pick $s^* \leftarrow \{0,1\}^\lambda$ and set $r^* = \mathsf{PRG}(a^* \oplus s^*)$.
2. Mask the challenge message as $c^* = m_b \oplus \mathsf{CWPRF.F}(\mathsf{fk}, (a^*, r^*, \tau^*))$.
3. Return the challenge ciphertext $\mathsf{ct}^* = (a^*, r^*, c^*)$ to $\mathcal{A}$.

The ciphertext distributions in both the hybrids are identical since by the correctness of $\mathsf{CWPRF.Eval}$ we have $\mathsf{CWPRF.Eval}(\mathsf{ek}, (a^*, r^*, \tau^*), s^*) = \mathsf{CWPRF.F}(\mathsf{fk}, (a^*, r^*, \tau^*))$. Therefore, Hybd_0 and Hybd_1 are identically distributed from $\mathcal{A}$'s view and we have $\Pr[\mathsf{H}_0] = \Pr[\mathsf{H}_1]$.

Hybd_2: This is same as hybrid 1 except that the challenger chooses r^* uniformly at random from $\{0,1\}^{2\lambda}$ instead of computing it as $r^* = \mathsf{PRG}(a^* \oplus s^*)$ for some $s^* \in \{0,1\}^\lambda$. The change is indicated below.

Challenge: The adversary $\mathcal{A}$ submits a challenge attribute $a^* \in \{0,1\}^\lambda$, a challenge tag $\tau^* \in 0,1^t$ and a pair of challenge messages $(m_0, m_1) \in \{0,1\}^\ell \times \{0,1\}^\ell$. The challenger proceeds as follows:

1. Set $r^* \leftarrow \{0,1\}^{2\lambda}$.
2. Mask the challenge message as $c^* = m_b \oplus \mathsf{CWPRF.F}(\mathsf{fk}, (a^*, r^*, \tau^*))$.
3. Return the challenge ciphertext $\mathsf{ct}^* = (a^*, r^*, c^*)$ to $\mathcal{A}$.

By the security of PRG (Def. 1), we have $|\Pr[\mathsf{H}_1] - \Pr[\mathsf{H}_2]| = \mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{B}}(\lambda) = negl(\lambda)$.

Hybd_3: This is exactly the same as Hybd_2 except that we set $c^* = m_b \oplus u^*$ where u^* is chosen uniformly at random from the co-domain $\mathcal{Y}$ of $\mathsf{CWPRF.F}(\mathsf{fk}, \cdot)$ instead of computing it as $c^* = m_b \oplus \mathsf{CWPRF.F}(\mathsf{fk}, (a^*, r^*, \tau^*))$ for some $r^* \in \{0,1\}^{2\lambda}$. We now show that hybrids 2 and 3 are indistinguishable for any PPT adversary $\mathcal{A}$ due to the $\mathsf{Adp\text{-}IND}$ security of CWPRF.

First, we note that the statement $(a^*, r^*, \tau^*) \notin L$ with overwhelming probability of $(1 - 2^{-\lambda})$ as r^* is picked uniformly at random from $\{0,1\}^{2\lambda}$. Secondly, we only need to compute constrained keys $\mathsf{fk}_{C_{f,\tau}} \leftarrow \mathsf{CWPRF.ConKey}(\mathsf{fk}, C_{f,\tau})$ for circuits $C_{f,\tau}$ such that either $f(a^*) = 0$ or $\tau \neq \tau^*$. Therefore, $C_{f,\tau}(a^*, r^*, \tau^*) = 0$

for all $f \in Q_f$ (which are due to the secret-key queries of $\mathcal{A}$) and for all $\tau \in Q_\tau$ (which corresponds to the decryption queries of $\mathcal{A}$).

Suppose $\mathcal{B}_1$ is a CWPRF adversary against the Adp-IND security. Then, $\mathcal{B}_1$ can perfectly simulates $\mathcal{A}$ as follows. Upon receiving the public evaluation key ek from the CWPRF challenger, $\mathcal{B}_1$ uses its constrained key oracle to produce a secret-key $\mathsf{sk}_f = \mathsf{fk}_{C_{f,\tau}}$ for a query (f, τ) from $\mathcal{A}$. Similarly, for a decryption query (ct, f, τ), $\mathcal{B}_1$ again employs it's constrained key oracle to get a secret-key $\mathsf{sk}_f = \mathsf{fk}_{C_{f,\tau}}$ and then use it to decrypt ct following the standard procedure. At some point, when $\mathcal{B}_1$ receives a challenge tuple $(a^*, \tau^*, (m_0, m_1))$ from $\mathcal{A}$, the CWPRF-adversary $\mathcal{B}_1$ sends its challenge statement as (a^*, r^*, τ^*) for some $r^* \leftarrow \{0,1\}^{2\lambda}$ and receives a pseudorandom value u^* from the CWPRF-challenger. Now, $\mathcal{B}_1$ chooses a random bit b and transfers the challenge ciphertext $\mathsf{ct}^* = (a^*, r^*, m_b \oplus u^*)$ to $\mathcal{A}$. Note that, $\mathcal{A}$ can still continue querying for secret-keys and decryptions to which $\mathcal{B}_1$ proceeds as before.

It is easy to see that if $u^* = \mathsf{CWPRF.F}(\mathsf{fk}, (a^*, r^*, \tau^*))$ then $\mathcal{B}_1$ simulates hybrid 2 and if $u^* \leftarrow \mathcal{Y}$ then $\mathcal{B}_1$ simulates hybrid 3. Moreover, $\mathcal{B}_1$ is a legitimate adversary since $(a^*, r^*, \tau^*) \notin L$ with high probability and $\mathcal{B}_1$ is able to execute the experiment with constrained keys $\mathsf{fk}_{C_{f,\tau}}$ satisfying $C_{f,\tau}(a^*, r^*, \tau^*) = 0$ for all $f \in Q_f$ and $\tau \in Q_\tau$. Therefore, if $\mathcal{A}$ can distinguish between hybrids 2 and 3 with a non-negligible advantage then $\mathcal{B}_1$ breaks the Adp-IND security of CWPRF. Hence, we have $|\Pr[\mathsf{H}_2] - \Pr[\mathsf{H}_3]| = \mathsf{Adv}^{\mathsf{Adp\text{-}IND}}_{\mathcal{B}_1,\mathsf{CWPRF}}(\lambda) = negl(\lambda)$.

Finally, we note that in hybrid 3 the challenge bit b is statistically hidden inside $c^* = m_b \oplus u^*$ since u^* is chosen as uniformly at random from $\mathcal{Y}$. This ensures that $\mathcal{A}$'s advantage in guessing the bit b in hybrid 3 is at most $\frac{1}{2}$ even when it has access to the secret-key and decryption oracles. This completes the proof of Adp-INDCCA security of our ABE.