A Punctured Programming Approach to Adaptively Secure Functional Encryption

Brent Waters*
University of Texas at Austin
bwaters@cs.utexas.edu

Abstract

We propose a new construction for achieving adaptively secure functional encryption for poly-sized circuits from indistinguishability obfuscation. Our reduction has polynomial loss to the underlying primitives. We develop a "punctured programming" approach to constructing and proving systems where outside of obfuscation we rely only on primitives constructable from pseudo random generators.

*Brent Waters is supported by NSF CNS-0915361 and CNS-0952692, CNS-1228599 DARPA through the U.S. Office of Naval Research under Contract N00014-11-1-0382, DARPA N11AP20006, Google Faculty Research award, the Alfred P. Sloan Fellowship, Microsoft Faculty Fellowship, and Packard Foundation Fellowship.

1 Introduction

In traditional encryption systems a message, m, is encrypted to a particular user's public key PK. Later a user that holds the corresponding secret key will be able to decrypt the ciphertext and learn the contents of the message. At the same time any computationally bounded attacker will be unable to get any additional information on the message.

While this communication paradigm is appropriate for many scenarios such as targeted sharing between users, there exists many applications that demand a more nuanced approach to sharing encrypted data. For example, suppose that an organization encrypts video surveillance images and stores these ciphertexts in a large online database. Later, we would like to give an analyst the ability to view all images that match a particular pattern such as ones that include a facial image that pattern matches with an particular individual. In a traditional encryptions system we would be forced to either give the analyst the secret key enabling them to view everything or give them nothing and no help at all.

The concept of functional encryption (FE) was proposed to move beyond this all or nothing view of decryption. In a functional encryption system a secret key SK_f is associated with a function f. When a user attempts to decrypt a ciphertext CT encrypted for message m with secret key SK_f, he will learn $f(m)$. The security of functional encryption states that an attacker that receives keys for any polynomial number of functions $f_1, \ldots, f_Q$ should not be able to distinguish between an encryption of m_0, m_1 as long as $\forall i \; f_i(m_0) = f_i(m_1)$.

The concept of functional encryption first appeared under the guise of predicate encryption [BW07, KSW08] with the nomenclature later being updated[SW08, BSW11] to functional encryption. In addition, functional encryption has early roots in Attribute-Based Encryption [SW05] and searching on encrypted data [BCOP04].

A central challenge is to achieve functional encryption for as expressive functionality classes as possible — ideally one would like to achieve it for any poly-time computable function. Until recently, the best available was roughly limited to the inner product functionality proposed by Katz, Sahai, and Waters [KSW08]. This state of affairs changed dramatically with the introduction of a candidate indistinguishability obfuscation [BGI$^+$12] system for all poly-size circuits by Garg, Gentry, Halevi, Raykova, Sahai, and Waters [GGH$^+$13]. The authors showed that a function encryption system for any poly-sized circuits can be built from an indistinguishability obfuscator plus public key encryption and statistically simulation sound non-interactive zero knowledge proofs.

Thinking of Adaptive Security While the jump from inner product functionality to any poly-size circuit is quite significant, one limitation of the GGHRSW functional encryption system is that it only offers a *selective* proof of security where the attacker must declare the challenge messages before seeing the parameters of the FE system. Subsequently, Boyle, Chung and Pass [BCP14] proposed an FE construction based on an obfuscator that is differing inputs secure. We briefly recall that an obfuscator $\mathcal{O}$ is indistinguishability secure if it is computationally difficult for an attacker to distinguish between obfuscations $\mathcal{O}(C_0)$ and $\mathcal{O}(C_1)$ for any two (similar sized) circuits that are functionally equivalent (i.e. $\forall x \; C_0(x) = C_1(x)$). On the other hand assuming an obfuscator is differing inputs [BCP14, ABG$^+$13] secure allows for C_0 and C_1 to not be functionally equivalent, but requires that for any PPT attacker that distinguishes between obfuscations of the two circuits there must a PPT extraction algorithm that finds some x such that $C_0(x) \neq C_1(x)$. Thus, differing inputs obfuscation is in a qualitatively different class of "knowledge definitions". Furthermore, there is significant evidence [GGHW14] that there exists certain functionalities with auxiliary input that are impossible to build obfuscate under the differing inputs definition.

Our goal is to build adaptively secure functional encryption systems from indistinguishability obfuscation. We require that our reductions have polynomial loss of security relative to the underlying primitives. In addition, we want to take a minimalist approach to the primitives we utilize outside of obfuscation. In particular, we wish to avoid the use of additional "strong tools" such as non-interactive zero knowledge proofs or additional assumptions over algebraic groups.

Our Results In this work we propose two new constructions for achieving secure functional encryption (for poly-sized circuits) from indistinguishability obfuscation. We develop a "punctured programming" approach [SW14] to constructing and proving systems where our main tools in addition to obfuscation are a selectively secure puncturable pseudo random functions. We emphasize puncturable PRFs are themselves constructible from pseudo random generators [GGM84, BW13, BGI13, KPTZ13].

We start toward our an FE construction which is provably secure against any attacker that is limited to making all of its private key queries *after* it sees the challenge ciphertext.[1] While this is attacker is still restricted relative to a fully adaptive attacker, we observe that such a definition is already stronger than the commonly used selective restriction.

To build our system we first introduce an abstraction that we call puncturable deterministic encryption (PDE). The main purpose of this abstraction is to serve in some places as a slightly higher level and more convenient abstraction to work with than puncturable PRFs. A PDE system is a symmetric key and deterministic encryption scheme and consists of four algorithms: $\mathsf{Setup}_{\mathsf{PDE}}(1^\lambda)$, $\mathsf{Encrypt}_{\mathsf{PDE}}(K, m)$, $\mathsf{Decrypt}_{\mathsf{PDE}}(K, \mathrm{CT})$, and $\mathsf{Puncture}_{\mathsf{PDE}}(K, m_0, m_1)$. The first three algorithms have the usual correctness semantics. The fourth puncture algorithm takes as input a master key and two messages (m_0, m_1) and outputs a punctured key that can decrypt all ciphertexts except for those encrypted for either of the two messages — recall encryption is deterministic so there are only two such ciphertexts. The security property of PDE is stated as a game where the attacker gives two messages (m_0, m_1) to the attacker and then returns back a punctured key as well as two ciphertexts, one encrypted under each message. In a secure system no PPT attacker will be able to distinguish which ciphertext is associated with which message.

Our PDE encryption mechanism is rather simple and is derived from the hidden trigger mechanism from the Sahai-Waters [SW14] deniable encryption scheme. PDE Ciphertexts are of the form:

$$\mathrm{CT} = (A = F_1(K_1, m), \quad B = F_2(K_2, A) \oplus m).$$

where F_1 and F_2 are puncturable pseudo random functions, with F_1 being an injective function. Decryption requires first computing $m' = B \oplus F_2(K_2, A)$ and then checking that $F_1(K_1, m') = A$. [2]

With this tool in place we are now ready to describe our first construction. The setup algorithm will first choose a puncturable PRF key K for function F. Next, it will create the public parameters PP as an obfuscation of a program called InitialEncrypt. The InitialEncrypt program will take in randomness r and compute a tag $t = \mathrm{PRG}(r)$. Then it will output a PDE key k which is derived from $F(K, t)$. The encryption algorithm can use this obfuscated program to encrypt as follows. It will simply choose a random value $r \in \{0,1\}^\lambda$, where λ is the security parameter. It then runs the obfuscated program on r to receive (t, k) and then creates the ciphertext CT as $(t, c = \mathsf{Encrypt}_{\mathsf{PDE}}(k, m))$.

The secret key SK_f for a function f will be created as an obfuscated program. This program will take as input a ciphertext $\mathrm{CT} = (t, c)$. The program first computes k from $F(K, t)$, then uses k to decrypt c to a message m and outputs $f(m)$. The decryption algorithm is simply to run the obfuscated program on the ciphertext.

The proof of security of our first system follows what we can a "key-programming" approach. The high level idea is that for each key we will hardwire in the decryption response into each secret key obfuscated program for when the input is the challenge ciphertext. For all other inputs the key computes decryption normally. Our key-programming approach is enabled by two important factors. First, in the security game there is a single challenge ciphertext so only one hardwiring needs to be done per key. Second, since all queries come after the challenge messages (m_0, m_1) are declared we will know where we need to puncture to create our hardwiring.

Intuitively, our proof can be broken down into two high level steps. First, we will perform a set of steps that allow us to hardwire the decryption answers to all of the secret keys for the challenge ciphertext. Next, we use PDE security to move from encrypting m_b for challenge bit $b \in \{0,1\}$ to always encrypting m_0— independent of the bit b. (The actual proof of Section 5 contains multiple hybrids and is more intricate.)

[1] This model has been called semi-adaptive in other contexts [CW14].

[2] Despite sharing the term deterministic, our security definition of PDEs does not have much in common with deterministic encryption [BFO08, BFOR08] which has a central goal of hiding information among message distributions of high entropy.

Handling Full Security We now move to dealing with full security where we need to handle private key queries on both sides of the challenge ciphertext. At this point it is clear that relying only key-programming will not suffice. First, a pre-challenge ciphertext key for function f will need to be created before the challenge messages (m_0, m_1) are declared, so it will not even be known at key creation time what $f(m_0) = f(m_1)$ will be. One natural thought is to try to program the answers to pre-challenge keys in the challenge ciphertext, however, this clearly cannot work since the ciphertext is of bounded size and we wish to support an unbounded number of queries. Another perhaps more promising direction would be to apply the dual system encryption [Wat09] methodology. While this methodology has been successful in proving adaptive security in ABE and simple predicate encryption schemes [LOS+10, OT10] there is not an obvious translation to this setting. However, we will see below that our techniques have connections to this methodology and could be considered a certain variant of it.

Our central idea is to replace the notion of key-programming with what we call "key-signaling". In a key-signaling system a normal ciphertext will be associated with a single message m which we refer to as an α-message. The decryption algorithm will use the secret key to prepare an α-signal for the ciphertext which will enable normal decryption. However, the ciphertext can also have a second form in which it is associated with two messages m_α and m_β. The underlying semantics are that if it receives an α-signal it uses m_α and if it receives a β-signal it uses m_β.

These added semantics open up new strategy for proving security. In the initial security game the challenge ciphertext encrypts m_b for challenge bit b. It will only receive α-signals from keys. Next we (indistinguishably) move the challenge ciphertext to encrypt m_b as the α-message and m_0 as the β-message. All keys still send only α-signals. Now one by one we change each key to send an β-signal to the challenge ciphertext as opposed to an α-signal. This step is feasible since for any queried function f we must have that $f(m_b) = f(m_0)$. Finally, we are able to erase the message m_b since no key is signaling for it.

Stepping back we can see that instead of storing the response of decryption for the challenge ciphertext at each key, we are storing the fact that it is using the second message in decryption.

To actually execute this strategy we will need further ideas. The first thing that we do is associate each ciphertext with a random tag t (like in the first solution) and also each key with a random tag y. Second, both private keys and ciphertexts contain obfuscated programs. The decryption algorithm will use both the key program and ciphertext program to decrypt.

A key program will take as input the ciphertext tag t and generate two objects. The first is a PDE encryption, c, of the signal α. This PDE ciphertext is generated under a key derived from the *pair* of tags (t, y). Second it will create a *one-bounded* secure private key $\mathrm{SK}_{\mathrm{OB}}$ for functionality f. The master secret key for this one-bounded system is again derived from the pair of tags (t, y). Next, the ciphertext program will take as input the key tag y and PDE encryption c. It will first use the pair of keys to generate a PDE key which decrypts c. If the signal is correct, it then creates a one-bounded ciphertext $\mathrm{CT}_{\mathrm{OB}}$ of its message m. Finally, the decryption algorithm combines the one-bounded scheme and ciphertext to decrypt and learn the output.

Our technique using a signaling strategy to generate a bounded key and ciphertext pair for each tag pair. Gorbunov, Vaikuntanathan and Wee [GVW12] proved adaptive security of a public key FE 1-bounded scheme from IND-CPA secure public key encryption. Since we actually only need master key encryption, we observe that that this can be achieved from IND-CPA symmetric key encryption. Thus, we maintain our goal of not using heavy weight primitives outside of obfuscation. One important fact is that the GVW scheme is proven to be 1-bounded adaptively secure regardless of whether the private key query comes before or after the challenge ciphertext.

The actual proof of security requires several hybrid steps and we defer further details to Section 6.

2 Preliminaries

In this section, we define indistinguishability obfuscation, and puncturable pseudo random functions (PRFs). All the variants of PRFs that we consider can be constructed from one-way functions.

2.1 Indistinguishability Obfuscation and PRFs

The definition of indistinguishability obfuscation below is adapted from [GGH+13]; following [KSW14] the main difference with previous definitions is that we uncouple the security parameter from the circuit size by directly defining indistinguishability obfuscation for all circuits:

Definition 1 (Indistinguishability Obfuscator ($i\mathcal{O}$))**.** A uniform PPT machine $i\mathcal{O}$ is called an *indistinguishability obfuscator* for circuits if the following conditions are satisfied:

- For all security parameters $\lambda \in \mathbb{N}$, for all circuits C, for all inputs x, we have that

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

- For any (not necessarily uniform) PPT adversaries *Samp*, D, there exists a negligible function α such that the following holds: if $\Pr[|C_0| = |C_1| \text{ and } \forall x, C_0(x) = C_1(x) : (C_0, C_1, \sigma) \leftarrow Samp(1^\lambda)] > 1-\alpha(\lambda)$, then we have:

$$\Big| \Pr\left[D(\sigma, i\mathcal{O}(\lambda, C_0)) = 1 : (C_0, C_1, \sigma) \leftarrow Samp(1^\lambda)\right] \\ - \Pr\left[D(\sigma, i\mathcal{O}(\lambda, C_1)) = 1 : (C_0, C_1, \sigma) \leftarrow Samp(1^\lambda)\right]\Big| \leq \alpha(\lambda)$$

Such indistinguishability obfuscators for circuits were constructed under novel algebraic hardness assumptions in [GGH+13].

2.2 Puncturable PRFs

Puncturable PRFs were defined by Sahai and Waters [SW14] as a simple type of constrained PRF [BW13, BGI13, KPTZ13]. They define a *puncturable* PRFs as a PRF for which a key can be given out that allows evaluation of the PRF on all inputs, except for a designated polynomial-size set of inputs.

Definition 2. A *puncturable* family of PRFs F mapping is given by a triple of Turing Machines (Key_F, $\mathrm{Puncture}_F$, and Eval_F), and a pair of computable functions $n(\cdot)$ and $m(\cdot)$, satisfying the following conditions:

- **[Functionality preserved under puncturing]** For every PPT adversary A such that $A(1^\lambda)$ outputs a set $S \subseteq \{0,1\}^{n(\lambda)}$, then for all $x \in \{0,1\}^{n(\lambda)}$ where $x \notin S$, we have that:

$$\Pr\left[\mathsf{Eval}_F(K, x) = \mathsf{Eval}_F(K_S, x) : K \leftarrow \mathrm{Key}_F(1^\lambda), K_S = \mathrm{Puncture}_F(K, S)\right] = 1$$

- **[Pseudorandom at punctured points]** For every PPT adversary (A_1, A_2) such that $A_1(1^\lambda)$ outputs a set $S \subseteq \{0,1\}^{n(\lambda)}$ and state σ, consider an experiment where $K \leftarrow \mathrm{Key}_F(1^\lambda)$ and $K_S = \mathrm{Puncture}_F(K, S)$. Then we have

$$\left| \Pr\left[A_2(\sigma, K_S, S, \mathsf{Eval}_F(K, S)) = 1\right] - \Pr\left[A_2(\sigma, K_S, S, U_{m(\lambda)\cdot|S|}) = 1\right] \right| = negl(\lambda)$$

where $\mathsf{Eval}_F(K, S)$ denotes the concatenation of $\mathsf{Eval}_F(K, x_1)), \ldots, \mathsf{Eval}_F(K, x_k))$ where $S = \{x_1, \ldots, x_k\}$ is the enumeration of the elements of S in lexicographic order, $negl(\cdot)$ is a negligible function, and U_ℓ denotes the uniform distribution over ℓ bits.

For ease of notation, we write $F(K, x)$ to represent $\mathsf{Eval}_F(K, x)$. We also represent the punctured key $\text{Puncture}_F(K, S)$ by $K(S)$.

The GGM tree-based construction of PRFs [GGM84] from one-way functions are easily seen to yield puncturable PRFs where the punctured size sizes are polynomial in the size of the set S, as recently observed by [BW13, BGI13, KPTZ13]. Thus we have:

Theorem 1. [GGM84, BW13, BGI13, KPTZ13] If one-way functions exist, then for all efficiently computable functions $n(\lambda)$ and $m(\lambda)$, there exists a puncturable PRF family that maps $n(\lambda)$ bits to $m(\lambda)$ bits.

Next we consider families of PRFs that are with high probability injective using the definition:

Definition 3. A *statistically injective* (puncturable) PRF family with failure probability $\epsilon(\cdot)$ is a family of (puncturable) PRFs F such that with probability $1 - \epsilon(\lambda)$ over the random choice of key $K \leftarrow \text{Key}_F(1^\lambda)$, we have that $F(K, \cdot)$ is injective.

If the failure probability function $\epsilon(\cdot)$ is not specified, then $\epsilon(\cdot)$ is a negligible function.

Theorem 2. If one-way functions exist, then for all efficiently computable functions $n(\lambda)$, $m(\lambda)$, and $e(\lambda)$ such that $m(\lambda) \geq 2n(\lambda) + e(\lambda)$, there exists a puncturable statistically injective PRF family with failure probability $2^{-e(\lambda)}$ that maps $n(\lambda)$ bits to $m(\lambda)$ bits.

The proof of this theorem is contained in Sahai-Waters [SW14].

Sampling Master Keys At times instead of running the $\text{Key}_F(1^\lambda)$ algorithm to generate the master key for a puncturable PRF we will generate the master key by simply sampling a uniformly random string $K \in \{0,1\}^\lambda$ where λ is the security parameter. We argue that we can do this without loss of generality. First, suppose there exists a puncturable PRF system as defined above. Then we can create another puncturable PRF system which uses the random coins, r used in $\text{Key}_F(1^\lambda; r)$ as the master secret key. Since the original master secret key can be generated from these coins, we can adapt the algorithms to use r as the master secret key — any algorithm that needs to use the original master key K will simply first generate it by calling $\text{Key}_F(1^\lambda; r)$. Second, suppose there is an algorithm that chooses a master secret key as a random string of length $z > \lambda$. Then we can always create another puncturable PRF system with length λ secret keys that simply using a pseudo random generator to expand the key from λ to z bits.

The above transformations are standard observations used in cryptography. We mention this here since in our constructions we will often sample a directly as a random string instead of going through the process of picking randomness and then generating a key from the randomness. The reason is simply to cut down on the number of steps we need in our exposition.

3 Functional Encryption

Our syntax for functional encryption roughly follows in the line of Boneh-Sahai-Waters [BSW11] except we specialize our notation for the case where the private key is a function f and the ciphertext input is a message m. This is without loss of generality when f can be any poly-sized circuit and thus includes a universal circuit.

For security we use the indistinguishability notion, which was the first one considered for functional encryption (as well as predicate encryption [BW07, KSW08]). De Caro et. al. [CJO+13] show how in the randomoracle model one can transform a system with indistinguishability secure into one with strong simulation security.

Definition 4 (Functional Encryption)**.** A *functional encryption scheme* for a class of functions $\mathcal{F} = \mathcal{F}(\lambda)$ over message space $\mathcal{M} = \mathcal{M}(\lambda)$ consists of four algorithms $\mathcal{FE} = \{\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}\}$:

$\mathsf{Setup}(1^\lambda)$ – a polynomial time algorithm that takes the unary representation of the security parameter λ and outputs public parameters PP and a master secret key MSK.

$\mathsf{KeyGen}(\mathrm{MSK}, f)$ – a polynomial time algorithm that takes as input the master secret key MSK and a description of function $f \in \mathcal{F}$ and outputs a corresponding secret key SK_f.

$\mathsf{Encrypt}(\mathrm{PP}, x)$ – a polynomial time algorithm that takes the public parameters PP and a string x and outputs a ciphertext CT.

$\mathsf{Decrypt}(\mathrm{SK}_f, \mathrm{CT})$ – a polynomial time algorithm that takes a secret key SK_f and ciphertext encrypting message $m \in \mathcal{M}$ and outputs $f(m)$.

A functional encryption scheme is correct for $\mathcal{F}$ if for all $f \in \mathcal{F}$ and all messages $m \in \mathcal{M}$:

$$\Pr[\ (\mathrm{PP}, \mathrm{MSK}) \leftarrow \mathsf{Setup}(\mathbf{1}^\lambda);\ \ \mathsf{Decrypt}(\mathsf{KeyGen}(\mathrm{MSK}, f), \mathsf{Encrypt}(\mathrm{PP}, m)) \neq f(m)\ ] = negl(\lambda).$$

Indistinguishability Security for Functional Encryption

We describe indistinguishability security as a multi-phased game between an attacker $\mathcal{A}$ and a challenger.

Setup: The challengers runs $(\mathrm{PP}, \mathrm{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$ and gives PP to $\mathcal{A}$.

Query Phase 1: $\mathcal{A}$ adaptively submits queries f in $\mathcal{F}$ and is given $\mathrm{SK}_f \leftarrow \mathsf{KeyGen}(\mathrm{MSK}, f)$. This step can be repeated any polynomial number of times by the attacker.

Challenge: $\mathcal{A}$ submits two messages $m_0, m_1 \in \mathcal{M}$ such that $f(m_0) = f_(m_1)$ for all functions f queried in the key query phase. The challenger then samples $\mathrm{CT}^* \leftarrow \mathsf{Encrypt}(\mathrm{PP}, m_b)$ for the attacker.

Query Phase 2: $\mathcal{A}$ continues to issue key queries as before subject to the restriction that any f queried must satisfy $f(m_0) = f(m_1)$.

Guess: $\mathcal{A}$ eventually outputs a bit b' in $\{0, 1\}$.

The advantage of an algorithm $\mathcal{A}$ in this game is $\mathsf{Adv}_{\mathcal{A}} = \Pr[b' = b] - \frac{1}{2}$.

Definition 5. A functional encryption scheme is indistinguishability secure if for all poly-time $\mathcal{A}$ the function $\mathsf{Adv}_{\mathcal{A}}(\lambda)$ is negligible.

Definition 6. In the above security game we define a *post challenge ciphertext* attacker as one that does not make any key queries in Phase 1. We define a functional encryption scheme to be *post challenge ciphertext* indistinguishability secure if for any poly-time algorithm $\mathcal{A}$ that is a post challenge ciphertext attacker the advantage of $\mathcal{A}$ is negligible in the indistinguishability security game.

Remark 1. We remark that any system that is *post challenge ciphertext* secure must also be selectively secure. Recall that a selective attacker is required to give the challenge messages m_0, m_1 before seeing the parameters and then can make as many queries as desired. We first observe that if there exists a selective attacker on a system that makes both Phase 1 and Phase 2 queries, then there exists a selective attacker that makes only Phase 2 queries. Intuitively, even though a selective attacker it can then make both Phase 1 and Phase 2 queries, the abilitiy to make Phase 1 queries does not provide any additional leverage (over only making Phase 2 queries) since the selective committed to the challenge messages. With this observation in mind we now see that a post challenge ciphertext attacker has the same power except it has the additional leverage in that it can delay its decision of what challenge messages to commit to until after seeing the public parameters.

3.1 One Bounded FE schemes

In our main construction we will use as a building block adaptively secure one-bounded secure functional encryption schemes. These are functional encryption schemes in which the attacker is allowed to make at most one private key query. Gorbunov, Vaikuntanathan and Wee [GVW12] proved adaptive security of a 1-bounded scheme from IND-CPA secure public key encryption. (They later use this to build k-bounded schemes for larger k.) Following Sahai and Seyalioglu [SS10], they base their construction off of Yao garbled circuits. An important point is that the 1-bounded security holds whether they key query comes before or after the challenge ciphertext.

For our purposes we will only need a one-bounded FE scheme with symmetric key (or master key) encryption. This is clearly implied by a public key FE scheme. In this security model an attacker will not be given any public parameters, but can query an encryption oracle a polynomial number of times. Without loss of generality we will sometimes assume that the master secret key is chosen as a uniform string of λ bits for security parameter λ. (See the previous section's discussion on sampling master keys.) We observe that the 1-bounded GVW scheme can by based off of IND-CPA security of symmetric key encryption if the one-bounded FE scheme itself is only required to provide for master key encryption.

4 Puncturable Deterministic Encryption

In this section we define a primitive of puncturable deterministic encryption and show how to build it from (injective) puncturable PRFs. The main purpose of this abstraction is to give a slightly higher level tool (relative to puncturable PRFs) to work with in our punctured programming construction and proofs.

4.1 Definition

Definition 7 (Puncturable Deterministic Encryption)**.** A *puncturable deterministic encryption* (PDE) scheme is defined over a message space $\mathcal{M} = \mathcal{M}(\lambda)$ and consists of four algorithms: (possibly) randomized algorithms $\mathsf{Setup}_{\mathsf{PDE}}$, and $\mathsf{Puncture}_{\mathsf{PDE}}$ along with deterministic algorithms $\mathsf{Encrypt}_{\mathsf{PDE}}$ and $\mathsf{Decrypt}_{\mathsf{PDE}}$. All algorithms will be poly-time in the security parameter.

$\mathsf{Setup}_{\mathsf{PDE}}(1^\lambda)$ The setup algorithm takes a security parameter and uses its random coins to generate a key K from a keyspace $\mathcal{K}$.

$\mathsf{Encrypt}_{\mathsf{PDE}}(K, m)$ The encrypt algorithm takes as input a key K and a message m. It outputs a ciphertext CT. *The algorithm is deterministic.*

$\mathsf{Decrypt}_{\mathsf{PDE}}(K, \mathrm{CT})$ The decrypt algorithm takes as input a key K and ciphertext CT. It outputs either a message $m \in \mathcal{M}$ or a special reject symbol $\perp$.

$\mathsf{Puncture}_{\mathsf{PDE}}(K, m_0, m_1)$ The puncture algorithm takes as input a key $K \in \mathcal{K}$ as well as two messages m_0, m_1. It creates and outputs a new key $K(m_0, m_1) \in \mathcal{K}$. The parentheses are used to syntactically indicate what is punctured.

Correctness A punctured deterministic encryption scheme is correct if there exists a negligible function negl such that the following holds for all λ and all pairs of messages $m_0, m_1 \in \mathcal{M}(\lambda)$.

Let $K = \mathsf{Setup}_{\mathsf{PDE}}(1^\lambda)$ and $K(m_0, m_1) \leftarrow \mathsf{Puncture}_{\mathsf{PDE}}(K, m_0, m_1)$. Then for all $m \neq m_0, m_1$

$$\Pr[\mathsf{Decrypt}_{\mathsf{PDE}}(K(m_0, m_1), \mathsf{Encrypt}_{\mathsf{PDE}}(K, m)) \neq m] = \mathrm{negl}(\lambda).$$

In addition, we have that for all m (including m_0, m_1)

$$\Pr[\mathsf{Decrypt}_{\mathsf{PDE}}(K, \mathsf{Encrypt}_{\mathsf{PDE}}(K, m)) \neq m] = \mathrm{negl}(\lambda).$$

Definition 8. We say that a correct scheme is perfectly correct if the above probability is 0 and otherwise say that it is statistically correct.

(Selective) Indistinguishability Security for Punctured Deterministic Encryption

We describe indistinguishability security as a multi-phased game between an attacker $\mathcal{A}$ and a challenger.

Setup: The attacker selects two messages $m_0, m_1 \in \mathcal{M}$ and sends these to the challenger. The challenger runs $K = \mathsf{Setup}_{\mathsf{PDE}}(1^\lambda)$ and $K(m_0, m_1) = \mathsf{Puncture}_{\mathsf{PDE}}(K, m_0, m_1)$. It then chooses a random bit $b \in \{0,1\}$ and computes

$$T_0 = \mathsf{Encrypt}_{\mathsf{PDE}}(K, m_b),\ T_1 = \mathsf{Encrypt}_{\mathsf{PDE}}(K, m_{1-b}).$$

It gives the punctured key $K(m_0, m_1)$ as well as T_0, T_1 to the attacker.

Guess: $\mathcal{A}$ outputs a bit b' in $\{0,1\}$.

The advantage of an algorithm $\mathcal{A}$ in this game is $\mathsf{Adv}_{\mathcal{A}} = \Pr[b' = b] - \frac{1}{2}$.

Definition 9. A puncturable deterministic encryption scheme is indistinguishability secure if for all poly-time $\mathcal{A}$ the function $\mathsf{Adv}_{\mathcal{A}}(\lambda)$ is negligible.

One can also consider an adaptive game of security where the attacker can probe an encryption oracle on multiple messages before committing to m_0, m_1. However, we do not explore this further in this paper.

Remark 2. Our definition allows for a key to be punctured at two messages. One possibility is to extend this abstraction to allow for puncturing at many messages (and likewise adapt the security game). However, we chose a narrower definition since it is simpler and sufficient to suit our purposes.

Sampling Master Keys At times instead of running the $\mathsf{Setup}_{\mathsf{PDE}}(1^\lambda)$ algorithm to generate the master key for a puncturable PRF we will generate the master key by simply sampling a uniformly random string $k \in \{0,1\}^\lambda$ where λ is the security parameter. We can also do this without loss of generality from an argument similar to the one we gave for sampling puncturable PRF keys (see Section 2.2).Our motivation again is to cut down on the description length of our primitives and proofs.

4.2 A Construction from Puncturable PRFs

We now describe our PDE construction which is derived from the mechanism used to implement "hidden triggers" in the Sahai-Waters [SW14] deniable encryption system. The PDE scheme we provide is parameterized over a security parameter λ and has message space $\mathcal{M} = \mathcal{M}(\lambda) = \{0,1\}^\lambda$. It makes use off two puncturable PRF families. The first is a statistically injective puncturable PRF F_1 that takes inputs from λ bits to $\ell = \ell(\lambda)$ bits and the second F_2 goes from ℓ bits to λ bits.

$\mathsf{Setup}_{\mathsf{PDE}}(1^\lambda)$ The setup algorithm samples keys $K_1 \leftarrow \mathrm{Key}_{F_1}(1^\lambda)$ and $K_1 \leftarrow \mathrm{Key}_{F_2}(1^\lambda)$.

$\mathsf{Encrypt}_{\mathsf{PDE}}(K = (K_1, K_2), m)$ The encryption algorithm (deterministically) computes a ciphertext as:

$$\mathrm{CT} = (A = F_1(K_1, m), \quad B = F_2(K_2, A) \oplus m).$$

$\mathsf{Decrypt}_{\mathsf{PDE}}(K, \mathrm{CT} = (A, B))$ The decryption algorithm first computes $m' = F_2(K_2, A) \oplus B$. Next, it checks if $F_1(K_1, m') \stackrel{?}{=} A$. If so, it outputs m', otherwise it outputs $\perp$.

$\mathsf{Puncture}_{\mathsf{PDE}}(K, m_0, m_1)$ The algorithm computes $d_A = F_1(K_1, m_0)$ and $e_A = F_1(K_1, m_1)$. It sets $K_1(m_0, m_1) = \mathrm{Puncture}_{F_1}(K, \{m_0, m_1\})$ and $K_2(d_A, e_A) = \mathrm{Puncture}_{F_2}(K, \{d_A, e_A\})$. [3] The output PDE key is $K(m_0, m_1) = (K_1(m_0, m_1), K_2(d_A, e_A))$.

[3]We assume that the distribution of $\mathrm{Puncture}_{F_1}(K_1, \{m_0, m_1\})$ is the same as $\mathrm{Puncture}_{F_1}(K_1, \{m_1, m_0\})$. This can be easily achieved by treating the parameters in lexicographic or random order. (We also assume this for F_2.)

Correctness. Correctness holds in the case where K_1 is sampled such that the function $F(K_1, \cdot)$ is injective. For use of non-punctured keys, correctness follows from observation. The encryption and decryption algorithms can also be used on punctured keys . Here correctness will follow for key $K(m_0, m_1)$ on all messages $m \neq m_0, m_1$ as long as $F_1(K_1, m) \neq F_1(K_1, m_0)$ or $F_1(K_1, m_1)$. This bad event will not occur as long as $F(K_1, \cdot)$ is injective.

4.2.1 Security

We sketch a proof of security via a sequence of hybrid games.

Game 1

1. Attacker declares (m_0, m_1) and challenger samples $K = (K_1, K_2)$.
2. Challenger computes $d_A = F_1(K_1, m_0), e_A = F_1(K_1, m_0)$.
3. Challenger computes $d_B = F_2(K_2, d_A), e_B = F_2(K_2, e_A)$.
4. Challenger outputs $K(m_0, m_1) = (K_1(m_0, m_1), K_2(d_A, e_A))$.
5. Challenge flips a coin b and outputs: $T_b = (d_A, d_B \oplus m_0) \quad T_{1-b} = (e_A, e_B \oplus m_1)$.
6. Attacker guesses b' and wins if $b = b'$.

Game 2
Line 2. Challenger chooses random d_A, e_A.
By punctured PRF security no attacker can distinguish Game 1 and Game 2.

Game 3
Line 3. Challenger chooses random d_B, e_B.
By punctured PRF security no attacker can distinguish Game 2 and Game 3.

However, we can now see that since d_A, d_B, e_A, e_B are all chosen uniformly at random in Game 3 we have that T_0, T_1 information theoretically hide the bit b. This final information theoretic argument depends on the fact that the distribution of $\text{Puncture}_{F_1}(K_1, \{m_0, m_1\})$ is the same as $\text{Puncture}_{F_1}(K_1, \{m_1, m_0\})$.

5 A Post Challenge Ciphertext Secure Construction

We now describe our construction for a functional encryption (FE) scheme that is post challenge ciphertext secure. We let the message space $\mathcal{M} = \mathcal{M}(\lambda) = \{0,1\}^{\ell(\lambda)}$ for some polynomial function ℓ and the function class be $\mathcal{F} = \mathcal{F}(\lambda)$.

We will use a puncturable PRF $F(\cdot, \cdot)$ such that when we fix the key K we have that $F(K, \cdot)$ takes in a 2λ bit input and outputs λ bits. In addition, we use a puncturable deterministic encryption scheme (PDE) where the message space $\mathcal{M}$ is the same as that of the (FE) system. In our PDE systems master keys are sampled uniformly at random from $\{0,1\}^\lambda$. Finally, we use an indistinguishability secure obfuscator and a length doubling pseudo random generator PRG : $\{0,1\}^\lambda \to \{0,1\}^{2\lambda}$.

Setup(1^λ)
The setup algorithm first chooses a random punctured PRF key $K \leftarrow \text{Key}_F(1^\lambda)$ and sets this as the master secret key MSK. Next it creates an obfuscation of the program Initial-Encrypt as $P \leftarrow i\mathcal{O}(1^\lambda, \text{Initial-Encrypt:1}[K])$.[4] This obfuscated program, P, serves as the public parameters PP.

[4]The program Initial-Encrypt:1 is padded to be the same size as Initial-Encrypt:2.

$\mathsf{Encrypt}(\mathrm{PP} = P(\cdot), m \in \mathcal{M})$
The encryption algorithm chooses random $r \in \{0,1\}^\lambda$. It then runs the obfuscated program P on r to get:

$$(t, k) \leftarrow P(r).$$

It then computes $\mathsf{Encrypt}_{\mathsf{PDE}}(k, m) = c$. The output ciphertext is $\mathrm{CT} = (t, c)$.

$\mathsf{KeyGen}(\mathrm{MSK}, f \in \mathcal{F}(\lambda))$ The KeyGen algorithm produces an obfuscated program P_f by obfuscating

$$P_f \leftarrow i\mathcal{O}(\textsc{Key-Eval}{:}1[K, f]).^5$$

$\mathsf{Decrypt}(\mathrm{CT} = (t, c), \mathrm{SK} = P_f)$ The decryption algorithm takes as input a ciphertext CT and a secret key SK which is an obfuscated program P_f. It runs $P_f(t, c)$ and outputs the response.

Correctness Correctness follows in a rather straightforward manner from the correctness of the underlying primitives. We briefly sketch the correctness argument. Suppose we call the encryption algorithm for message m with randomness r. The obfuscated program generates $(t, k) = (\mathrm{PRG}(r), F(K, t))$. Then it creates the ciphertext $\mathrm{CT} = (t, c = \mathsf{Encrypt}_{\mathsf{PDE}}(k, m))$. Now let's examine what occurs when $\mathsf{Decrypt}(\mathrm{CT} = (t, c), \mathrm{SK}_f = P_f)$ is called where P_f was a secret key created from function f. The decryption algorithm calls $P_f(t, c)$. The (obfuscated) program will compute the same PDE key $k = F(K, t)$ as used to create the ciphertext. Then it will use the PDE decryption algorithm and obtain m. This follows via the correctness of the PDE scheme. Finally, it outputs $f(m)$ which is the correct output.

Initial-Encrypt:1

Constants: Puncturable PRF key K.
Input: Randomness $r \in \{0,1\}^\lambda$.

1. Let $t = \mathrm{PRG}(r)$.
2. Compute: $k = F(K, t)$.
3. Output: (t, k).

Figure 1: Program Initial-Encrypt:1

Key-Eval:1

Constants: PRF key K, function description $f \in \mathcal{F}$.
Input: (t, c).

1. Compute: $k = F(K, t)$.
2. Output $f(\mathsf{Decrypt}_{\mathsf{PDE}}(k, c))$. (If $\mathsf{Decrypt}_{\mathsf{PDE}}(k, c)$ evaluates to $\perp$ the program outputs $\perp$.)

Figure 2: Program Key-Eval:1

5.1 Proof of Security

Theorem 3. *The above functional encryption scheme is post challenge ciphertext secure if it is instantiated with a secure punctured PRF, puncturable deterministic encryption scheme, pseudo random generator, and indistinguishability secure obfuscator.*

[5] The program $\textsc{Key-Eval}{:}1$ is padded to be the same size as $\textsc{Key-Eval}{:}2$.

Proof. To prove the above theorem, we first define a sequence of games where the first game is the original FE security game. Then we show (based on the security of different primitives) that any poly-time attacker's advantage in each game must be negligibly close to that of the previous game. We begin by with describing Game 1 in detail, which is the (post challenge ciphertext) FE security game instantiated with our construction. From there we describe the sequence of games where each game is described by its modification from the previous game. We continue to enumerate each step in every game in order to ease verification of our lemmas.

Game 1 The first game is the original security game instantiated for our construction.

1. Challenger computes $K \leftarrow \text{Key}_F(1^\lambda)$ and randomly chooses the challenge bit $b \in \{0,1\}$.
2. Challenger chooses random $r^* \in \{0,1\}^\lambda$ and computes $t^* = \text{PRG}(r^*)$.
3. Challenger creates $P \leftarrow i\mathcal{O}(1^\lambda, \textsc{Initial-Encrypt}{:}1[K])$ and passes P to attacker.
4. Attacker gives messages $m_0, m_1 \in \mathcal{M}$ to challenger. (No Phase 1 queries.)
5. Challenger computes $k^* \leftarrow F(K, t^*)$.
6. Challenger computes $c^* \leftarrow \mathsf{Encrypt}_{\mathsf{PDE}}(k^*, m_b)$ and outputs challenge ciphertext as $\text{CT}^* = (t^*, c^*)$.
7. On attacker key query for function $f \in \mathcal{F}$ the challenger responds with $P_f \leftarrow i\mathcal{O}(\textsc{Key-Eval}{:}1[K, f])$.
8. The attacker gives a bit b' and wins if $b' = b$.

Game 2

1. Challenger computes $K \leftarrow \text{Key}_F(1^\lambda)$ and randomly chooses the challenge bit $b \in \{0,1\}$.
2. <u>Challenger chooses random $t^* \in \{0,1\}^{2\lambda}$.</u>
3. Challenger creates $P \leftarrow i\mathcal{O}(1^\lambda, \textsc{Initial-Encrypt}{:}1[K])$ and passes P to attacker.
4. Attacker gives messages $m_0, m_1 \in \mathcal{M}$ to challenger. (No Phase 1 queries.)
5. Challenger computes $k^* \leftarrow F(K, t^*)$.
6. Challenger computes $c^* \leftarrow \mathsf{Encrypt}_{\mathsf{PDE}}(k^*, m_b)$ and outputs challenge ciphertext as $\text{CT}^* = (t^*, c^*)$.
7. On attacker key query for function $f \in \mathcal{F}$ the challenger responds with $P_f \leftarrow i\mathcal{O}(\textsc{Key-Eval}{:}1[K, f])$.
8. The attacker gives a bit b' and wins if $b' = b$.

Game 3 Parameters program is now punctured at t^*.

1. Challenger computes $K \leftarrow \text{Key}_F(1^\lambda)$ and randomly chooses the challenge bit $b \in \{0,1\}$.
2. Challenger chooses random $t^* \in \{0,1\}^{2\lambda}$ <u>and computes computes $K(t^*) = \text{Puncture}_F(K, t^*)$.</u>
3. <u>Challenger creates $P \leftarrow i\mathcal{O}(1^\lambda, \textsc{Initial-Encrypt}{:}2[K(t^*)])$ and passes P to attacker.</u>
4. Attacker gives messages $m_0, m_1 \in \mathcal{M}$ to challenger. (No Phase 1 queries.)
5. Challenger computes $k^* \leftarrow F(K, t^*)$.
6. Challenger computes $c^* \leftarrow \mathsf{Encrypt}_{\mathsf{PDE}}(k^*, m_b)$ and outputs challenge ciphertext as $\text{CT}^* = (t^*, c^*)$.
7. On attacker key query for function $f \in \mathcal{F}$ the challenger responds with $P_f \leftarrow i\mathcal{O}(\textsc{Key-Eval}{:}1[K, f])$.
8. The attacker gives a bit b' and wins if $b' = b$.

Game 4 Obfuscated programs in secret keys are punctured and hardwired.

1. Challenger computes $K \leftarrow \mathrm{Key}_F(1^\lambda)$ and randomly chooses the challenge bit $b \in \{0,1\}$.
2. Challenger chooses random $t^* \in \{0,1\}^{2\lambda}$ and computes computes $K(t^*) = \mathrm{Puncture}_F(K, t^*)$.
3. Challenger creates $P \leftarrow i\mathcal{O}(1^\lambda, \textsc{Initial-Encrypt}{:}2[K(t^*)])$ and passes P to attacker.
4. Attacker gives messages $m_0, m_1 \in \mathcal{M}$ to challenger. (No Phase 1 queries.)
5. Challenger computes $k^* \leftarrow F(K, t^*)$.
6. Challenger computes $c^* \leftarrow \mathsf{Encrypt}_{\mathsf{PDE}}(k^*, m_b)$ and outputs challenge ciphertext as $\mathrm{CT}^* = (t^*, c^*)$.
7. Let $k' = \mathsf{Puncture}_{\mathsf{PDE}}(k^*, m_0, m_1)$. Choose random $\gamma \in \{0,1\}$ and let $c_0 = \mathsf{Encrypt}_{\mathsf{PDE}}(k^*, m_\gamma)$ and let $c_1 = \mathsf{Encrypt}_{\mathsf{PDE}}(k^*, m_{1-\gamma})$.
   Consider each attacker key query for function $f \in \mathcal{F}$. Let $d_f = f(m_0) = f(m_1)$. The challenger responds with $P_f \leftarrow i\mathcal{O}(\textsc{Key-Eval}{:}2[K(t^*), t^*, f, c_0, c_1, d_f, k'])$.
8. The attacker gives a bit b' and wins if $b' = b$.

Game 5

1. Challenger computes $K \leftarrow \mathrm{Key}_F(1^\lambda)$ and randomly chooses the challenge bit $b \in \{0,1\}$.
2. Challenger chooses random $t^* \in \{0,1\}^{2\lambda}$ and computes computes $K(t^*) = \mathrm{Puncture}_F(K, t^*)$.
3. Challenger creates $P \leftarrow i\mathcal{O}(1^\lambda, \textsc{Initial-Encrypt}{:}2[K(t^*)])$ and passes P to attacker.
4. Attacker gives messages $m_0, m_1 \in \mathcal{M}$ to challenger. (No Phase 1 queries.)
5. Challenger chooses random $k^* \in \{0,1\}^\lambda$.
6. Challenger computes $c^* \leftarrow \mathsf{Encrypt}_{\mathsf{PDE}}(k^*, m_b)$ and outputs challenge ciphertext as $\mathrm{CT}^* = (t^*, c^*)$.
7. Let $k' = \mathsf{Puncture}_{\mathsf{PDE}}(k^*, m_0, m_1)$. Choose random $\gamma \in \{0,1\}$ and let $c_0 = \mathsf{Encrypt}_{\mathsf{PDE}}(k^*, m_\gamma)$ and let $c_1 = \mathsf{Encrypt}_{\mathsf{PDE}}(k^*, m_{1-\gamma})$. Consider each attacker key query for function $f \in \mathcal{F}$. Let $d_f = f(m_0) = f(m_1)$. The challenger responds with $P_f \leftarrow i\mathcal{O}(\textsc{Key-Eval}{:}2[K(t^*), t^*, f, c_0, c_1, d_f, k'])$.
8. The attacker gives a bit b' and wins if $b' = b$.

Game 6

1. Challenger computes $K \leftarrow \mathrm{Key}_F(1^\lambda)$ and randomly chooses the challenge bit $b \in \{0,1\}$.
2. Challenger chooses random $t^* \in \{0,1\}^{2\lambda}$computes computes $K(t^*) = \mathrm{Puncture}_F(K, t^*)$.
3. Challenger creates $P \leftarrow i\mathcal{O}(1^\lambda, \textsc{Initial-Encrypt}{:}2[K(t^*)])$ and passes P to attacker.
4. Attacker gives messages $m_0, m_1 \in \mathcal{M}$ to challenger. (No Phase 1 queries.)
5. Challenger chooses random $k^* \in \{0,1\}^\lambda$.
6. Challenger computes $c^* \leftarrow \mathsf{Encrypt}_{\mathsf{PDE}}(k^*, m_0)$ and outputs challenge ciphertext as $\mathrm{CT}^* = (t^*, c^*)$.
7. Let $k' = \mathsf{Puncture}_{\mathsf{PDE}}(k^*, m_0, m_1)$. Choose random $\gamma \in \{0,1\}$ and let $c_0 = \mathsf{Encrypt}_{\mathsf{PDE}}(k^*, m_\gamma)$ and let $c_1 = \mathsf{Encrypt}_{\mathsf{PDE}}(k^*, m_{1-\gamma})$. Consider each attacker key query for function $f \in \mathcal{F}$. Let $d_f = f(m_0) = f(m_1)$. The challenger responds with $P_f \leftarrow i\mathcal{O}(\textsc{Key-Eval}{:}2[K(t^*), t^*, f, c_0, c_1, d_f, k'])$.
8. The attacker gives a bit b' and wins if $b' = b$.

Initial-Encrypt:2

Constants: Punctured PRF key $K(t^*)$.
Input: Randomness $r \in \{0,1\}^\lambda$.

1. Let $t = PRG(r)$.
2. Compute: $k = F(K(t^*), t)$.
3. Output: (t, k).

Figure 3: Program Initial-Encrypt:2

Key-Eval:2

Constants: PRF key $K(t^*)$, tag value $t^* \in \{0,1\}^{2\lambda}$, function description $f \in \mathcal{F}$, PDE ciphertexts c_0, c_1, $d_f \in \{0,1\}$, and punctured deterministic encryption key $k' = k_{t^*}(m_0, m_1)$.
Input: (t, c).

1. If $t = t^*$ AND $c \neq c_0, c_1$ output $f(\mathsf{Decrypt}_{\mathsf{PDE}}(k', c))$.
2. If $t = t^*$ AND ($c = c_0$ OR $c = c_1$) output d_f.
3. Otherwise compute: $k = F(K, t)$.
4. Output $f(\mathsf{Decrypt}_{\mathsf{PDE}}(k, c))$.

Figure 4: Program Key-Eval:2

We observe that in this final game the attacker has no information on the challenger's bit b since the game always just encrypts m_0.

We now move to establishing the lemmas that argue the attacker's advantage must be negligibly close between successive games. We let $\mathsf{Adv}_{\mathcal{A},i}$ denote the advantage of algorithm $\mathcal{A}$ in Game i of guessing the bit b.

Lemma 1. If our pseudo random generator PRG is secure then for all PPT $\mathcal{A}$ we have that $\mathsf{Adv}_{\mathcal{A},1} - \mathsf{Adv}_{\mathcal{A},2} = \text{negl}(\lambda)$ for some negligible function negl.

Proof. We describe and analyze a PPT reduction algorithm $\mathcal{B}$ that plays the pseudo random generator security game. $\mathcal{B}$ first receives a PRG game challenge $T \in \{0,1\}^{2\lambda}$. It then runs the attacker and executes the security game as described in Game 1 with the exception that in Step 2 it lets $t^* = T$. If the attacker wins (i.e. $b' = b$), then $\mathcal{B}$ guesses '1' to indicated that T was in the image of a PRG; otherwise, it outputs '0' to indicate that T was chosen randomly.

We observe that when T is generated as $T = \text{PRG}(r)$, then $\mathcal{B}$ gives exactly the view of Game 1 to $\mathcal{A}$. Otherwise if T is chosen randomly the view is of Game 2. Therefore if $\mathsf{Adv}_{\mathcal{A},1} - \mathsf{Adv}_{\mathcal{A},2}$ is non-negligble, $\mathcal{B}$ must also have non-neglgible advantage against the pseudo random generator security game. ∎

Lemma 2. If $i\mathcal{O}$ is a secure indistinguishability obfuscator, then for all PPT $\mathcal{A}$ we have that $\mathsf{Adv}_{\mathcal{A},2} - \mathsf{Adv}_{\mathcal{A},3} = \text{negl}(\lambda)$ for some negligible function negl.

Proof. We describe and analyze a PPT reduction algorithm $\mathcal{B}$ that plays the indistinguishability obfuscation security game with $\mathcal{A}$. $\mathcal{B}$ runs steps 1-2 as in Game 2. Next it creates two circuits as $C_0 = \textsc{Initial-Encrypt}{:}1[K]$ and $C_1 = \textsc{Initial-Encrypt}{:}2[K(t^*)]$. It submits both of these to the IO challenger and receives back a program P which it passes to the attacker in step 3. It executes steps 4-8 as in Game 2. If the attacker wins (i.e. $b' = b$), then $\mathcal{B}$ guesses '0' to indicated that P was and obfuscation of C_0; otherwise, it guesses '1' to indicate it was an obfuscation of C_1.

We observe that when P is generated as and obfuscation of C_0, then $\mathcal{B}$ gives exactly the view of Game 2 to $\mathcal{A}$. Otherwise if P is chosen as an obfuscation of C_1 the view is of Game 2. In addition, the programs are functionally equivalent with all but negligible probability. The reason is that t^* is outside the image of the pseudo random generator with probability at least $1 - 2^{\lambda}$. Therefore if $\mathsf{Adv}_{\mathcal{A},2} - \mathsf{Adv}_{\mathcal{A},3}$ is non-negligble, $\mathcal{B}$ must also have non-neglgible advantage against the indisguishability obfuscation game. ■

Lemma 3. If $i\mathcal{O}$ is a secure indistinguishability obfuscator, then for all PPT $\mathcal{A}$ we have that $\mathsf{Adv}_{\mathcal{A},3} - \mathsf{Adv}_{\mathcal{A},4} = \text{negl}(\lambda)$ for some negligible function negl.

Proof. To prove this lemma we will consider a hybrid argument. Let $Q = Q(\lambda)$ be the number of private key queries issued by some attacker $\mathcal{A}$. (Without loss of generality we can assume $\mathcal{A}$ always makes exactly Q queries on every execution.) For $i \in [0, Q]$ we define $\mathsf{Game}\ 3, i$ to be the same as $\mathsf{Game}\ 3$ except that the first i private key queries of step 7 are handled as in $\mathsf{Game}\ 4$ and the last $Q - i$ are handled as in $\mathsf{Game}\ 3$. We observe that $\mathsf{Game}\ 3, 0$ is the same as $\mathsf{Game}\ 3$ and that $\mathsf{Game}\ 3, Q$ is the same as $\mathsf{Game}\ 4$. Thus to prove security we need to establish that no attacker can distinguish between $\mathsf{Game}\ 3, i$ and $\mathsf{Game}\ 3, i+1$ for $i \in [0, Q-1]$ with non negligible advantage.

We describe and analyze a PPT reduction algorithm $\mathcal{B}$ that plays the indistinguishability obfuscation security game with $\mathcal{A}$. $\mathcal{B}$ runs steps 1-6 as in $\mathsf{Game}\ 3$. For the first i queries of step 7 it answers as in $\mathsf{Game}\ 4$. For query $i + 1$ it creates two circuits as $C_0 = \text{Key-Eval:}1[K, f]$ where f is the function queried for. Next, let Let $k' = \mathsf{Puncture}_{\mathsf{PDE}}(k^*, m_0, m_1)$. Choose random $\gamma \in \{0, 1\}$ and let $c_0 = \mathsf{Encrypt}_{\mathsf{PDE}}(k^*, m_\gamma)$ and let $c_1 = \mathsf{Encrypt}_{\mathsf{PDE}}(k^*, m_{1-\gamma})$. Let $d_f = f(m_0) = f(m_1)$, where f is the key queried for. It creates $C_1 = \text{Key-Eval:}2[K(t^*), t^*, f, c_0, c_1, d_f, k']$.

It submits both of these to the IO challenger and receives back a program P which it passes to the attacker as P_f. It answers the rest of the queries of step 7 as in $\mathsf{Game}\ 3$ and completes step 8. If the attacker wins (i.e. $b' = b$), then $\mathcal{B}$ guesses '0' to indicated that P was and obfuscation of C_0; otherwise, it guesses '1' to indicate it was an obfuscation of C_1.

We observe that when P is generated as and obfuscation of C_0, then $\mathcal{B}$ gives exactly the view of $\mathsf{Game}\ 3, i$ to $\mathcal{A}$. Otherwise if P is chosen as an obfuscation of C_1 the view is of $\mathsf{Game}\ 3, i + 1$. In addition, the programs are functionally equivalent with all but negligible probability. The reason is that correctness holds for all messages with all but negligible probability. The only difference in the programs is that the response is hardwired in for two inputs. Therefore if $\mathsf{Adv}_{\mathcal{A},3,i} - \mathsf{Adv}_{\mathcal{A},3,i+1}$ is non-negligble, $\mathcal{B}$ must also have non-neglgible advantage against the indisguishability obfuscation game. ■

Lemma 4. If F is a selectively secure puncturable PRF then for all PPT $\mathcal{A}$ we have that $\mathsf{Adv}_{\mathcal{A},4} - \mathsf{Adv}_{\mathcal{A},5} = \text{negl}(\lambda)$ for some negligible function negl.

Proof. We describe and analyze a PPT reduction algorithm $\mathcal{B}$ that plays the selective puncturable PRF security game. $\mathcal{B}$ first runs step 1, then chooses t^* and submits it back to the punctured PRF challenger. It receives back a punctured key $K(t^*)$ and a challenge value z. It runs steps 3-4 for $\mathcal{A}$ as in $\mathsf{Game}\ 4$. In step 5 it sets $k^* = z$. It then runs step 6-8 as in $\mathsf{Game}\ 4$. We note that in step 7 the punctured key $K(t^*)$ is sufficient to create the challenge ciphertext and answer all private key queries. If the attacker wins (i.e. $b' = b$), then $\mathcal{B}$ guesses '1' to indicated that $z = F(K, t^*)$; otherwise, it outputs '0' to that z was chosen randomly.

We observe that when z is generated as $F(K, t^*)$, then $\mathcal{B}$ gives exactly the view of $\mathsf{Game}\ 4$ to $\mathcal{A}$. Otherwise if z is chosen randomly, the view is of $\mathsf{Game}\ 5$. Therefore if $\mathsf{Adv}_{\mathcal{A},4} - \mathsf{Adv}_{\mathcal{A},5}$ is non-negligble, $\mathcal{B}$ must also have non-neglgible advantage against the security of the puncturable PRF. ■

Lemma 5. If our puncturable deterministic encryption scheme is secure, then for all PPT $\mathcal{A}$ we have that $\mathsf{Adv}_{\mathcal{A},5} - \mathsf{Adv}_{\mathcal{A},6} = \text{negl}(\lambda)$ for some negligible function negl.

Proof. We begin by observing that the difference between $\mathsf{Game}\ 5$ and $\mathsf{Game}\ 6$ is that in $\mathsf{Game}\ 6$ the message encrypted in step 6 is always m_0 and in $\mathsf{Game}\ 5$ the message could be m_0 or m_1 depending on b. When the coin flip of $b = 0$ the views of the two games are identical. So if there is a difference in advantage it must solely concentrated in the case where we condition on $b = 1$ from step 1.

We describe and analyze a PPT reduction algorithm $\mathcal{B}$ that plays the puncturable deterministic encryption security game. $\mathcal{B}$ first executes steps 1-4 of Game 5, with the bit of step 1 being set to $b = 1$. Then it submits messages m_0, m_1 to the PDE challenger and receives back $k' = \mathsf{Puncture}_{\mathsf{PDE}}(k^*, m_0, m_1)$ and T_0, T_1. On step 6 it sets $c^* = T_0$.

It then runs the rest of the steps of Game 5. If the attacker wins (i.e. $b' = b$), then $\mathcal{B}$ guesses '1' to indicated that $c^* = T_1$ was an encryption of m_1; otherwise, it outputs '0' to that $c^* = T_1$ was an encryption of m_0. In step 7 it flips a bit γ' and lets $c_0 = T_{\gamma'}$ and $c_1 = T_{1-\gamma'}$. It then answers all queries as in Game 5. We note that k' was given from the challenger.

We observe that when $c^* = T_1$ is generated as $\mathsf{Encrypt}_{\mathsf{PDE}}(k^*, m_1)$ then $\mathcal{B}$ gives exactly the view of Game 5 (conditioned on $b = 1$ to $\mathcal{A}$. Otherwise if c^* is generated as $\mathsf{Encrypt}_{\mathsf{PDE}}(k^*, m_0)$ the view is of Game 6. Therefore if $\mathsf{Adv}_{\mathcal{A},5} - \mathsf{Adv}_{\mathcal{A},6}$ is non-negligble, $\mathcal{B}$ must also have non-neglgible advantage against the puncturable deterministic encryption system.

■

Now with all the lemmas in place we can pull our main theorem together. By a simple hybrid argument it follows that any PPT attacker's advantage in the original security Game 1 can be at most negligibly greater than its advantage in Game 6. However, the advantage of any attacker in Game 6 is 0 and thus the scheme is secure. ■

6 An Adaptively Secure Construction

We now describe our construction of a functional encryption (FE) scheme that is adaptively secure. We let the message space $\mathcal{M} = \{0,1\}^{\ell(\lambda)}$ for some polynomial function ℓ and the function class be $\mathcal{F}(\lambda) = \mathcal{F}$.

We will use two puncturable PRFs F_1, F_2 such that when we fix the keys K we have that $F_1(K, \cdot)$ takes in a 2λ bit input and outputs two bit strings of length λ and $F_2(K, \cdot)$ takes λ bits to five bitstrings of length λ. In addition, we use a puncturable deterministic encryption scheme where the message space is $\{0,1\}^{\lambda}$. In our Puncturable PRF and PDE systems master keys are sampled uniformly at random from $\{0,1\}^{\lambda}$. Finally, we use an indistinguishability secure obfuscator and an *injective* length doubling pseudo random generator $\mathrm{PRG} : \{0,1\}^{\lambda} \to \{0,1\}^{2\lambda}$.

Finally, we use a one-bounded secure functional encryption system with master key encryption consisting of algorithms: $\mathsf{KeyGenOB}, \mathsf{EncryptOB}, \mathsf{DecryptOB}$. We assume without loss of generality that the master key is chosen uniformly from $\{0,1\}^{\lambda}$. The message space $\mathcal{M}$ and key description space $f \in \mathcal{F}$ of the one bounded scheme is the same as the scheme we are constructing.

Setup(1^{λ})
The algorithm first chooses a random punctured PRF key $K \leftarrow \mathrm{Key}_{F_1}(1^{\lambda})$ which is set as the master secret key MSK. Next it creates an obfuscation of the program Initial-Encrypt as $P \leftarrow i\mathcal{O}(1^{\lambda}, \textsc{Initial-Encrypt}{:}1[K])$.[6]

Encrypt$(\mathrm{PP} = P(\cdot), m \in \mathcal{M})$
The encryption algorithm performs the following steps in sequence.

1. Chooses random $r \in \{0,1\}^{\lambda}$.
2. Sets $(t, K_t, \alpha) \leftarrow P(r)$.
3. Sets $\tilde{\alpha} = \mathrm{PRG}(\alpha)$.

[6]The program $\textsc{Initial-Encrypt}$:1 is padded to be the same size as $\textsc{Initial-Encrypt}$:2.) This obfuscated program, P serves as the public parameters PP.

4. Creates the program $C \leftarrow i\mathcal{O}(1^\lambda, \text{CT-E}\textsc{val}{:}1[K_t, \tilde{\alpha}, m])$.[7]

5. The output ciphertext is $\mathrm{CT} = (t, C)$.

KeyGen$(\mathrm{MSK}, f \in \mathcal{F}(\lambda))$
The KeyGen algorithm first chooses a random $y \in \{0,1\}^\lambda$. It next produces an obfuscated program P_f by obfuscating $P_f \leftarrow i\mathcal{O}(\textsc{Key-Signal}{:}1[K, f, y])$. [8]

The secret key is $\mathrm{SK} = (y, P_f)$.

Decrypt$(\mathrm{CT} = (t, C), \mathrm{SK} = (y, P_f))$
The decryption algorithm takes as input a ciphertext $\mathrm{CT} = (t, C)$ and a secret key $\mathrm{SK} = (y, P_f)$. It first computes $(a, \mathrm{SK}_{\mathrm{OB}}) = P_f(t)$. Next it computes $\mathrm{CT}_{\mathrm{OB}} = C(a, y)$. Finally, it will use the produced secret key to decrypt the produced ciphertext as $\mathsf{DecryptOB}(\mathrm{CT}_{\mathrm{OB}}, \mathrm{SK}_{\mathrm{OB}})$ and outputs the result.

Correctness We briefly sketch a correctness argument. Consider a ciphertext $\mathrm{CT} = (t, C)$ created for message m that is associated with tag t and a key for function f that is associated with tag y. On decryption the algorithm first calls $(a, \mathrm{SK}_{\mathrm{OB}}) = P_f(t)$. Here the obfuscated program computes: $(K_t, \alpha) = F_1(K, t)$, $(d, k, s_1, s_2, s_3) = F_2(K_t, y)$, and $a = \mathsf{Encrypt}_{\mathsf{PDE}}(d, \alpha)$ and $\mathrm{SK}_{\mathrm{OB}} = \mathsf{KeyGenOB}(k, f; s_2)$.

Next, it calls $\mathrm{CT}_{\mathrm{OB}} = C(a, y)$, where C was generated as an obfuscation of program $\text{CT-E}\textsc{val}{:}1[K_t, \tilde{\alpha}, m]$ where $\tilde{\alpha} = \mathrm{PRG}(\alpha)$. This obfuscated program will compute the same values of $(d, k, s_1, s_2, s_3) = F_2(K_t, y)$ as the key signal program. By correctness of the PDE system we will have that $\mathsf{Decrypt}_{\mathsf{PDE}}(d, a) = \alpha$ and thus the program will output $\mathsf{EncryptOB}(k, m; s_1)$. At this point the decryption algorithm has a one bounded private key for function f and a one bounded ciphertext for message m both created under the same master key k. Therefore, running the one-bounded decryption algorithm will produce $f(m)$.

Initial-Encrypt:1

Constants: Puncturable PRF key K.
Input: Randomness $r \in \{0,1\}^\lambda$.

1. Let $t = \mathrm{PRG}(r)$.
2. Compute $(K_t, \alpha) = F_1(K, t)$.
3. Output: (t, K_t, α).

Figure 5: Program Initial-Encrypt:1

CT-Eval:1

Constants: PRF key K_t, $\tilde{\alpha} \in \{0,1\}^{2\lambda}$, message $m \in \mathcal{M}$.
Input: PDE ciphertext a and value $y \in \{0,1\}^\lambda$.

1. Compute $(d, k, s_1, s_2, s_3) = F_2(K_t, y)$.
2. Compute $e = \mathsf{Decrypt}_{\mathsf{PDE}}(d, a)$.
3. If $\mathrm{PRG}(e) = \tilde{\alpha}$ output $\mathsf{EncryptOB}(k, m; s_1)$.
4. Else output a rejecting $\perp$.

Figure 6: Program CT-Eval:1

[7] The program CT-E$\textsc{val}$:1 is padded to be the same size as the maximum of CT-E$\textsc{val}$:2 and CT-E$\textsc{val}$:3.
[8] The program $\textsc{Key-Signal}$:1 is padded to be the same size as $\textsc{Key-Signal}$:2.

Key-Signal:1

Constants: PRF key K, function description $f \in \mathcal{F}$, tag $y \in \{0,1\}^\lambda$.
Input: $t \in \{0,1\}^{2\lambda}$.

1. Compute $(K_t, \alpha) = F_1(K, t)$.
2. Compute $(d, k, s_1, s_2, s_3) = F_2(K_t, y)$.
3. Compute and output $a = \mathsf{Encrypt}_{\mathsf{PDE}}(d, \alpha)$ and $\mathrm{SK}_{\mathrm{OB}} = \mathsf{KeyGenOB}(k, f; s_2)$.

Figure 7: Program Key-Signal:1

6.1 Proof of Security

Theorem 4. The above functional encryption scheme is adaptively secure if instantiated with a secure punctured PRF, puncturable deterministic encryption scheme, pseudo random generator, one-bounded functional encryption scheme and indistinguishability secure obfuscator.

To prove the above theorem, we first define a sequence of games where the first game is the original FE security game. Then we show (based on the security of different primitives) that any poly-time attacker's advantage in each game must be negligibly close to that of the previous game. We begin by with describing Game 1 in detail, which is the adaptive FE security game instantiated with our construction. From there we describe the sequence of games, where each game is described by its modification from the previous game. We continue to enumerate each step (in most descriptions) to ease verification of our claims.

Game 1 The first game is the original security game instantiated for our construction.

1. Challenger computes keys $K \leftarrow \mathrm{Key}_{F_1}(1^\lambda)$ and randomly chooses the challenge bit $b \in \{0,1\}$.
2. Challenger chooses random $r^* \in \{0,1\}^\lambda$ and computes $t^* = \mathrm{PRG}(r^*)$.
3. Challenger computes $K^*_{t^*}, \alpha^* = F_1(K, t^*)$.
4. Challenger sets $\tilde{\alpha^*} = \mathrm{PRG}(\alpha^*)$.
5. Challenger creates $P \leftarrow i\mathcal{O}(1^\lambda, \textsc{Initial-Encrypt}{:}1[K])$ and passes P to attacker.
6. Phase 1 Queries: Let f_j be the function of associated with the j-th query. Choose random $y_j \in \{0,1\}^\lambda$. Generate the j-th private key by computing $P_{f_j} \leftarrow i\mathcal{O}(\textsc{Key-Signal}{:}1[K, f_j, y_j])$. Output the key as (y_j, P_{f_j}).
7. Attacker gives messages $m_0, m_1 \in \mathcal{M}$ to challenger.
8. Challenger sets the program $C^* \leftarrow i\mathcal{O}(1^\lambda, \textsc{CT-Eval}{:}1[K^*_{t^*}, \tilde{\alpha^*}, m_b])$.
9. The output ciphertext is $\mathrm{CT} = (t^*, C^*)$.
10. Phase 2 Queries: Same as Phase 1 in step 6.
11. The attacker gives a bit b' and wins if $b' = b$.

Game 2

1. Challenger computes keys $K \leftarrow \mathrm{Key}_{F_1}(1^\lambda)$ and randomly chooses the challenge bit $b \in \{0,1\}$.
2. Challenger chooses random $t^* \in \{0,1\}^{2\lambda}$.
3. Challenger computes $K^*_{t^*}, \alpha^* = F_1(K, t^*)$.

4. Challenger sets $\tilde{\alpha^*} = \text{PRG}(\alpha^*)$.
5. Challenger creates $P \leftarrow i\mathcal{O}(1^\lambda, \text{Initial-Encrypt:}1[K])$ and passes P to attacker.
6. Phase 1 Queries: Let f_j be the function of associated with the j-th query. Choose random $y_j \in \{0,1\}^\lambda$. Generate the j-th private key by computing $P_{f_j} \leftarrow i\mathcal{O}(\text{Key-Signal:}1[K, f_j, y_j])$. Output the key as (y_j, P_{f_j}).
7. Attacker gives messages $m_0, m_1 \in \mathcal{M}$ to challenger.
8. Challenger sets the program $C^* \leftarrow i\mathcal{O}(1^\lambda, \text{CT-Eval:}1[K^*_{t^*}, \tilde{\alpha^*}, m_b])$.
9. The output ciphertext is $\text{CT} = (t^*, C^*)$.
10. Phase 2 Queries: Same as Phase 1 in step 6.
11. The attacker gives a bit b' and wins if $b' = b$.

Game 3

1. Challenger computes keys $K \leftarrow \text{Key}_{F_1}(1^\lambda)$ and randomly chooses the challenge bit $b \in \{0,1\}$.
2. Challenger chooses random $t^* \in \{0,1\}^{2\lambda}$ and computes computes $K(t^*) = \text{Puncture}_F(K, t^*)$.
3. Challenger computes $K^*_{t^*}, \alpha^* = F_1(K, t^*)$.
4. Challenger sets $\tilde{\alpha^*} = \text{PRG}(\alpha^*)$.
5. Challenger creates $P \leftarrow i\mathcal{O}(1^\lambda, \text{Initial-Encrypt:}2[K(t^*)])$ and passes P to attacker.
6. Phase 1 Queries: Let f_j be the function of associated with the j-th query. Choose random $y_j \in \{0,1\}^\lambda$. Generate the j-th private key by computing $P_{f_j} \leftarrow i\mathcal{O}(\text{Key-Signal:}1[K, f_j, y_j])$. Output the key as (y_j, P_{f_j}).
7. Attacker gives messages $m_0, m_1 \in \mathcal{M}$ to challenger.
8. Challenger sets the program $C^* \leftarrow i\mathcal{O}(1^\lambda, \text{CT-Eval:}1[K^*_{t^*}, \tilde{\alpha^*}, m_b])$.
9. The output ciphertext is $\text{CT} = (t^*, C^*)$.
10. Phase 2 Queries: Same as Phase 1 in step 6.
11. The attacker gives a bit b' and wins if $b' = b$.

Game 4

1. Challenger computes keys $K \leftarrow \text{Key}_{F_1}(1^\lambda)$ and randomly chooses the challenge bit $b \in \{0,1\}$.
2. Challenger chooses random $t^* \in \{0,1\}^{2\lambda}$ and computes computes $K(t^*) = \text{Puncture}_F(K, t^*)$.
3. Challenger computes $K^*_{t^*}, \alpha^* = F_1(K, t^*)$.
4. Challenger sets $\tilde{\alpha^*} = \text{PRG}(\alpha^*)$.
5. Challenger creates $P \leftarrow i\mathcal{O}(1^\lambda, \text{Initial-Encrypt:}2[K(t^*)])$ and passes P to attacker.
6. Phase 1 Queries: Let f_j be the function of associated with the j-th query.
   (a) Choose random $y_j \in \{0,1\}^\lambda$.
   (b) Compute $(d^*_j, k^*_j, s^*_{1,j}, s^*_{2,j}, s^*_{3,j}) = F_2(K_{t^*}, y_j)$.

(c) Compute $a_j^* = \mathsf{Encrypt}_{\mathsf{PDE}}(d_j^*, \alpha^*)$ and $\mathrm{SK}_{\mathrm{OB},j}^* = \mathsf{KeyGenOB}(k_j^*, f_j; s_{2,j}^*)$.
(d) Compute $P_{f_j} \leftarrow i\mathcal{O}(\textsc{Key-Signal}{:}2[K(t^*), t^*, a_j^*, \mathrm{SK}_{\mathrm{OB},j}^*, f_j, y_j])$.
(e) Output the key as (y_j, P_{f_j}).

7. Attacker gives messages $m_0, m_1 \in \mathcal{M}$ to challenger.
8. Challenger sets the program $C^* \leftarrow i\mathcal{O}(1^\lambda, \text{CT-}\textsc{Eval}{:}1[K_{t^*}^*, \tilde{\alpha^*}, m_b])$.
9. The output ciphertext is $\mathrm{CT} = (t^*, C^*)$.
10. Phase 2 Queries: Same as Phase 1 in step 6. (These are also changed as described above.)
11. The attacker gives a bit b' and wins if $b' = b$.

Game 5

1. Challenger computes keys $K \leftarrow \mathrm{Key}_{F_1}(1^\lambda)$ and randomly chooses the challenge bit $b \in \{0, 1\}$.
2. Challenger chooses random $t^* \in \{0, 1\}^{2\lambda}$ and computes computes $K(t^*) = \mathrm{Puncture}_F(K, t^*)$.
3. Challenger chooses random $K_{t^*}^*, \alpha^*$.
4. Challenger sets $\tilde{\alpha^*} = \mathrm{PRG}(\alpha^*)$.
5. Challenger creates $P \leftarrow i\mathcal{O}(1^\lambda, \textsc{Initial-Encrypt}{:}2[K(t^*)])$ and passes P to attacker.
6. Phase 1 Queries: Let f_j be the function of associated with the j-th query.
   (a) Choose random $y_j \in \{0, 1\}^\lambda$.
   (b) Compute $(d_j^*, k_j^*, s_{1,j}^*, s_{2,j}^*, s_{3,j}^*) = F_2(K_{t^*}, y_j)$.
   (c) Compute $a_j^* = \mathsf{Encrypt}_{\mathsf{PDE}}(d_j^*, \alpha^*)$ and $\mathrm{SK}_{\mathrm{OB},j}^* = \mathsf{KeyGenOB}(k_j^*, f_j; s_{2,j}^*)$.
   (d) Compute $P_{f_j} \leftarrow i\mathcal{O}(\textsc{Key-Signal}{:}2[K(t^*), t^*, a_j^*, \mathrm{SK}_{\mathrm{OB},j}^*, f_j, y_j])$.
   (e) Output the key as (y_j, P_{f_j}).
7. Attacker gives messages $m_0, m_1 \in \mathcal{M}$ to challenger.
8. Challenger sets the program $C^* \leftarrow i\mathcal{O}(1^\lambda, \text{CT-}\textsc{Eval}{:}1[K_{t^*}^*, \tilde{\alpha^*}, m_b])$.
9. The output ciphertext is $\mathrm{CT} = (t^*, C^*)$.
10. Phase 2 Queries: Same as Phase 1 in step 6.
11. The attacker gives a bit b' and wins if $b' = b$.

Game 6

1. Challenger computes keys $K \leftarrow \mathrm{Key}_{F_1}(1^\lambda)$ and randomly chooses the challenge bit $b \in \{0, 1\}$.
2. Challenger chooses random $t^* \in \{0, 1\}^{2\lambda}$computes computes $K(t^*) = \mathrm{Puncture}_F(K, t^*)$.
3. Challenger chooses random $K_{t^*}^*, \alpha^*$.
4. Challenger sets $\tilde{\alpha^*} = \mathrm{PRG}(\alpha^*)$ and chooses random $\tilde{\beta^*} \in \{0, 1\}^{2\lambda}$.
5. Challenger creates $P \leftarrow i\mathcal{O}(1^\lambda, \textsc{Initial-Encrypt}{:}2[K(t^*)])$ and passes P to attacker.
6. Phase 1 Queries: Let f_j be the function of associated with the j-th query.

(a) Choose random $y_j \in \{0,1\}^\lambda$.
(b) Compute $(d_j^*, k_j^*, s_{1,j}^*, s_{2,j}^*, s_{3,j}^*) = F_2(K_{t^*}, y_j)$.
(c) Compute $a_j^* = \mathsf{Encrypt}_{\mathsf{PDE}}(d_j^*, \alpha^*)$ and $\mathrm{SK}_{\mathrm{OB},j}^* = \mathsf{KeyGenOB}(k_j^*, f_j; s_{2,j}^*)$.
(d) Compute $P_{f_j} \leftarrow i\mathcal{O}(\textsc{Key-Signal}{:}2[K(t^*), t^*, a_j^*, \mathrm{SK}_{\mathrm{OB},j}^*, f_j, y_j])$.
(e) Output the key as (y_j, P_{f_j}).

7. Attacker gives messages $m_0, m_1 \in \mathcal{M}$ to challenger.
8. Challenger sets the program $C^* \leftarrow i\mathcal{O}(1^\lambda, \text{CT-}\textsc{Eval}{:}2[K_{t^*}^*, \tilde{\alpha^*}, \tilde{\beta^*}, m_b, m_0])$.
9. The output ciphertext is $\mathrm{CT} = (t^*, C^*)$.
10. Phase 2 Queries: Same as Phase 1 in step 6.
11. The attacker gives a bit b' and wins if $b' = b$.

$\mathsf{Game}\ 7$ Same as $\mathsf{Game}\ 6$ except we change line 4 to:

4. Challenger sets $\tilde{\alpha^*} = \mathrm{PRG}(\alpha^*)$, chooses $\beta^* \in \{0,1\}^\lambda$ at random and sets $\tilde{\beta^*} = \mathrm{PRG}(\beta^*)$.

$\mathsf{Game}\ 8, i$

1. Challenger computes keys $K \leftarrow \mathrm{Key}_{F_1}(1^\lambda)$ and randomly chooses the challenge bit $b \in \{0,1\}$.
2. Challenger chooses random $t^* \in \{0,1\}^{2\lambda}$ and computes computes $K(t^*) = \mathrm{Puncture}_F(K, t^*)$.
3. Challenger chooses random $K_{t^*}^*, \alpha^*$.
4. Challenger sets $\tilde{\alpha^*} = \mathrm{PRG}(\alpha^*)$, chooses $\beta^* \in \{0,1\}^\lambda$ at random and sets $\tilde{\beta^*} = \mathrm{PRG}(\beta^*)$.
5. Challenger creates $P \leftarrow i\mathcal{O}(1^\lambda, \textsc{Initial-Encrypt}{:}2[K(t^*)])$ and passes P to attacker.
6. Phase 1 Queries: Let f_j be the function of associated with the j-th query.
   (a) Choose random $y_j \in \{0,1\}^\lambda$.
   (b) Compute $(d_j^*, k_j^*, s_{1,j}^*, s_{2,j}^*, s_{3,j}^*) = F_2(K_{t^*}, y_j)$.
   (c) If $j > i$ then set $a_j^* = \mathsf{Encrypt}_{\mathsf{PDE}}(d_j^*, \alpha^*)$; otherwise if $j \le i$ set $a_j^* = \mathsf{Encrypt}_{\mathsf{PDE}}(d_j^*, \beta^*)$. Let $\mathrm{SK}_{\mathrm{OB},j}^* = \mathsf{KeyGenOB}(k_j^*, f_j; s_{2,j}^*)$.
   (d) Compute $P_{f_j} \leftarrow i\mathcal{O}(\textsc{Key-Signal}{:}2[K(t^*), t^*, a_j^*, \mathrm{SK}_{\mathrm{OB},j}^*, f_j, y_j])$.
   (e) Output the key as (y_j, P_{f_j}).
7. Attacker gives messages $m_0, m_1 \in \mathcal{M}$ to challenger.
8. Challenger sets the program $C^* \leftarrow i\mathcal{O}(1^\lambda, \text{CT-}\textsc{Eval}{:}2[K_{t^*}^*, \tilde{\alpha^*}, \tilde{\beta^*}, m_b, m_0])$.
9. The output ciphertext is $\mathrm{CT} = (t^*, C^*)$.
10. Phase 2 Queries: Same as Phase 1 in step 6.
11. The attacker gives a bit b' and wins if $b' = b$.

Game 9

1. Challenger computes keys $K \leftarrow \text{Key}_{F_1}(1^\lambda)$ and randomly chooses the challenge bit $b \in \{0,1\}$.
2. Challenger chooses random $t^* \in \{0,1\}^{2\lambda}$ and computes computes $K(t^*) = \text{Puncture}_F(K, t^*)$.
3. Challenger chooses random $K^*_{t^*}, \alpha^*$.
4. Challenger chooses $\tilde{\alpha}^* \in \{0,1\}^{2\lambda}$ at random, chooses $\beta^* \in \{0,1\}^\lambda$ at random and sets $\tilde{\beta^*} = \text{PRG}(\beta^*)$.
5. Challenger creates $P \leftarrow i\mathcal{O}(1^\lambda, \textsc{Initial-Encrypt}{:}2[K(t^*)])$ and passes P to attacker.
6. Phase 1 Queries: Let f_j be the function of associated with the j-th query.
   (a) Choose random $y_j \in \{0,1\}^\lambda$.
   (b) Compute $(d^*_j, k^*_j, s^*_{1,j}, s^*_{2,j}, s^*_{3,j}) = F_2(K_{t^*}, y_j)$.
   (c) Set $a^*_j = \mathsf{Encrypt}_{\mathsf{PDE}}(d^*_j, \beta^*)$. Let $\text{SK}^*_{\text{OB},j} = \mathsf{KeyGenOB}(k^*_j, f_j; s^*_{2,j})$.
   (d) Compute $P_{f_j} \leftarrow i\mathcal{O}(\textsc{Key-Signal}{:}2[K(t^*), t^*, a^*_j, \text{SK}^*_{\text{OB},j}, f_j, y_j])$.
   (e) Output the key as (y_j, P_{f_j}).
7. Attacker gives messages $m_0, m_1 \in \mathcal{M}$ to challenger.
8. Challenger sets the program $C^* \leftarrow i\mathcal{O}(1^\lambda, \text{CT-}\textsc{Eval}{:}2[K^*_{t^*}, \tilde{\alpha^*}, \tilde{\beta^*}, m_b, m_0])$.
9. The output ciphertext is $\text{CT} = (t^*, C^*)$.
10. Phase 2 Queries: Same as Phase 1 in step 6.
11. The attacker gives a bit b' and wins if $b' = b$.

Game 10 Is the same as Game 9 except line 8 is set to:
8. Challenger sets the program $C^* \leftarrow i\mathcal{O}(1^\lambda, \text{CT-}\textsc{Eval}{:}1[K^*_{t^*}, \tilde{\beta^*}, m_0])$.

We observe at this stage the interaction with the challenger is completely independent of b — note the message m_0 is encrypted regardless of b — and thus the attacker's advantage is 0 in this final game.

Initial-Encrypt:2

Constants: Puncturable PRF key $K(t^*)$.
Input: Randomness r.

1. Let $t = \text{PRG}(r)$.
2. Compute $(K_t, \alpha) = F_1(K(t^*), t)$.
3. Output: (t, K_t, α).

Figure 8: Program Initial-Encrypt:2

6.2 Indistinguishability Proofs Between Games

We now establish via a sequence of lemmas that the difference of the attacker's advantage between each adjacent game is neglgible. Most of the lemmas of indistinguishability are straightforward once the hybrid games are laid out. The exception is in proving the indistinguishability of Game $8, i$ from Game $8, i+1$. We handle this separately in its own subsection.

We let $\mathsf{Adv}_{\mathcal{A},i} = \Pr[b' = b] - \frac{1}{2}$ denote the advantage of algorithm $\mathcal{A}$ in Game i of guessing the bit b.

Key-Signal:2

Constants: PRF key $K(t^*)$, t^*, a^*, $\mathrm{SK}^*_{\mathrm{OB}}$, function description f, tag $y \in \{0,1\}^\lambda$.
Input: $t \in \{0,1\}^{2\lambda}$.

1. If $t = t^*$ output $a^*, \mathrm{SK}^*_{\mathrm{OB}}$.
2. Compute $(K_t, \alpha) = F_1(K(t^*), t)$.
3. Compute $(d, k, s_1, s_2, s_3) = F_2(K_t, y)$.
4. Compute and output $a = \mathsf{Encrypt}_{\mathsf{PDE}}(d, \alpha)$ and $\mathrm{SK}_{\mathrm{OB}} = \mathsf{KeyGenOB}(k, f; s_2)$

Figure 9: Program Key-Signal:2

CT-Eval:2

Constants: PRF key K_t, $\tilde{\alpha}, \tilde{\beta} \in \{0,1\}^{2\cdot\lambda}$, messages $m, m_{\mathrm{fixed}} \in \mathcal{M}$.
Input: (a, y).

1. Compute $(d, k, s_1, s_2, s_3) = F_2(K_t, y)$.
2. Compute $e = \mathsf{Decrypt}_{\mathsf{PDE}}(d, a)$.
3. If $\mathrm{PRG}(e) = \tilde{\alpha}$ output $\mathsf{EncryptOB}(k, m; s_1)$.
4. If $\mathrm{PRG}(e) = \tilde{\beta}$ output $\mathsf{EncryptOB}(k, m; s_3)$.
5. Else output a rejecting $\perp$.

Figure 10: Program CT-Eval:2

Lemma 6. If our pseudo random generator PRG is secure then for all PPT $\mathcal{A}$ we have that $\mathsf{Adv}_{\mathcal{A},1} - \mathsf{Adv}_{\mathcal{A},2} = \mathrm{negl}(\lambda)$ for some negligible function negl.

Proof. We describe and analyze a PPT reduction algorithm $\mathcal{B}$ that plays the pseudo random generator security game. $\mathcal{B}$ first receives a PRG game challenge $T \in \{0,1\}^{2\lambda}$. It then runs the attacker and runs the security game as described in Game 1 with the exception that in step 2 it lets $t^* = T$. If the attacker wins (i.e. $b' = b$), then $\mathcal{B}$ guesses '1' to indicate that T was in the image of $\mathrm{PRG}(\cdot)$; otherwise, it outputs '0' to that T was chosen randomly.

We observe that when T is generated as $T = \mathrm{PRG}(r)$, then $\mathcal{B}$ gives exactly the view of Game 1 to $\mathcal{A}$. Otherwise if T is chosen randomly the view is of Game 2. Therefore if $\mathsf{Adv}_{\mathcal{A},1} - \mathsf{Adv}_{\mathcal{A},2}$ is non-negligble, $\mathcal{B}$ must also have non-neglgible advantage against the pseudo random generator. ∎

Lemma 7. If $i\mathcal{O}$ is a secure indistinguishability obfuscator then for all PPT $\mathcal{A}$ we have that $\mathsf{Adv}_{\mathcal{A},2} - \mathsf{Adv}_{\mathcal{A},3} = \mathrm{negl}(\lambda)$ for some negligible function negl.

Proof. We describe and analyze a PPT reduction algorithm $\mathcal{B}$ that plays the indistinguishability obfuscation security game with $\mathcal{A}$. $\mathcal{B}$ runs steps 1-4 as in Game 2. Next it creates two circuits as $C_0 = \textsc{Initial-Encrypt}{:}1[K]$ and $C_1 = \textsc{Initial-Encrypt}{:}2[K(t^*)]$. It submits both of these to the IO challenger and receives back a program P which it passes to the attacker in step 5. It executes steps 6-10 as in Game 2. If the attacker wins (i.e. $b' = b$), then $\mathcal{B}$ guesses '0' to indicate that P was and obfuscation of C_0; otherwise, it guesses '1' to indicate it was an obfuscation of C_1.

We observe that when P is generated as and obfuscation of C_0, then $\mathcal{B}$ gives exactly the view of Game 2 to $\mathcal{A}$. Otherwise if P is chosen as an obfuscation of C_1 the view is of Game 2. In addition, the programs are functionally equivalent with all but negligible probability. The reason is that t^* is outside the image of the pseudo random generator with probability at least $1 - 2^\lambda$. Therefore if $\mathsf{Adv}_{\mathcal{A},2} - \mathsf{Adv}_{\mathcal{A},3}$ is non-negligble, $\mathcal{B}$ must also have non-neglgible advantage against the indisguishability obfuscation game. ∎

Lemma 8. If $i\mathcal{O}$ is a secure indistinguishability obfuscator then for all PPT $\mathcal{A}$ we have that $\mathsf{Adv}_{\mathcal{A},3} - \mathsf{Adv}_{\mathcal{A},4} = \text{negl}(\lambda)$ for some negligible function negl.

Proof. To prove this claim we will consider a hybrid argument. Let $Q = Q(\lambda)$ be the number of private key queries issued by some attacker $\mathcal{A}$. These include both Phase 1 and Phase 2 queries — that is the number of Phase 1 plus Phase 2 queries sums to Q. For $i \in [0, Q]$ we define $\mathsf{Game}\ 3, i$ to be the same as $\mathsf{Game}\ 3$ except that the first i private key queries are handled as in $\mathsf{Game}\ 4$ and the last $Q - i$ are handled as in $\mathsf{Game}\ 3$. We observe that $\mathsf{Game}\ 3, 0$ is the same as $\mathsf{Game}\ 3$ and that $\mathsf{Game}\ 3, Q$ is the same as $\mathsf{Game}\ 4$. Thus, to prove security we need to establish that no attacker can distinguish between $\mathsf{Game}\ 3, i$ and $\mathsf{Game}\ 3, i+1$ for $i \in [0, Q-1]$ with non negligible advantage.

We describe and analyze a PPT reduction algorithm $\mathcal{B}$ that plays the indistinguishability obfuscation security game with $\mathcal{A}$. $\mathcal{B}$ runs steps 1-5 as in $\mathsf{Game}\ 3$. For the first i private key queries it answers as in $\mathsf{Game}\ 4$. Let f be the function associated with the $i+1$-th key query. For query $i+1$ it will creates two circuits. The first is created as $C_0 = \text{Key-Signal:}1[K, f_{i+1}, y_{i+1}]$. Next, it computes $(d^*_{i+1}, k^*_{i+1}, s^*_{1,i+1}, s^*_{2,i+1}, s^*_{3,i+1}) = F_2(K_{t^*}, y_{i+1})$, $a^*_{i+1} = \mathsf{Encrypt}_{\mathsf{PDE}}(d^*_{i+1}, \alpha^*)$ and $\mathrm{SK}^*_{\mathrm{OB},i+1} = \mathsf{KeyGenOB}(k^*_{i+1}, f_{i+1}; s^*_{2,i+1})$. Then in creates the second circuit as $C_1 = \text{Key-Signal:}2[K(t^*), t^*, a^*_{i+1}, \mathrm{SK}^*_{\mathrm{OB},i+1}, f_{i+1}, y_{i+1}])$.

It submits both of these to the IO challenger and receives back a program P which it passes to the attacker as P_f. It answers the rest of the queries of step 6 as in $\mathsf{Game}\ 3$ and completes steps 7-9 and 11. If the attacker wins (i.e. $b' = b$), then $\mathcal{B}$ guesses '0' to indicate that P was and obfuscation of C_0; otherwise, it guesses '1' to indicate it was an obfuscation of C_1.

We observe that when P is generated as and obfuscation of C_0, then $\mathcal{B}$ gives exactly the view of $\mathsf{Game}\ 3, i$ to $\mathcal{A}$. Otherwise if P is chosen as an obfuscation of C_1 the view is of $\mathsf{Game}\ 3, i+1$. In addition, the programs are functionally equivalent with all but negligible probability. The only difference in the programs is that the response is hardwired in for one input. Therefore if $\mathsf{Adv}_{\mathcal{A},3,i} - \mathsf{Adv}_{\mathcal{A},3,i+1}$ is non-negligble, $\mathcal{B}$ must also have non-neglgible advantage against the indisguishability obfuscation game. ■

Lemma 9. If F is a selectively secure puncturable PRF then for all PPT $\mathcal{A}$ we have that $\mathsf{Adv}_{\mathcal{A},4} - \mathsf{Adv}_{\mathcal{A},5} = \text{negl}(\lambda)$ for some negligible function negl.

Proof. We describe and analyze a PPT reduction algorithm $\mathcal{B}$ that plays the selective puncturable PRF security game. $\mathcal{B}$ first runs step 1. In step 2 it chooses t^* and submits it to the punctured PRF challenger. It receives back a punctured key $K(t^*)$ and a challenge value (z_1, z_2). In step 3 it sets $(K^*_{t^*}, \alpha^*) = (z_1, z_2)$. It then runs step 4-10 as in $\mathsf{Game}\ 4$. We note that in steps 5 and 6 that the punctured key $K_1(t^*)$ is sufficient to create all programs. If the attacker wins (i.e. $b' = b$), then $\mathcal{B}$ guesses '1' to indicate that $(z_1, z_2) = F(K, t^*)$; otherwise, it outputs '0' to that (z_1, z_2) was chosen randomly.

We observe that when (z_1, z_2) is generated as $F(K, t^*)$, then $\mathcal{B}$ gives exactly the view of $\mathsf{Game}\ 4$ to $\mathcal{A}$. Otherwise if z is chosen randomly the view is of $\mathsf{Game}\ 4'$. Therefore if $\mathsf{Adv}_{\mathcal{A},4} - \mathsf{Adv}_{\mathcal{A},5}$ is non-negligble, $\mathcal{B}$ must also have non-neglgible advantage against the security of the puncturable PRF.

■

Lemma 10. If $i\mathcal{O}$ is a secure indistinguishability obfuscator then for all PPT $\mathcal{A}$ we have that $\mathsf{Adv}_{\mathcal{A},5} - \mathsf{Adv}_{\mathcal{A},6} = \text{negl}(\lambda)$ for some negligible function negl.

Proof. We describe and analyze a PPT reduction algorithm $\mathcal{B}$ that plays the indistinguishability obfuscation security game with $\mathcal{A}$. $\mathcal{B}$ runs steps 1-7 as in $\mathsf{Game}\ 6$. Next it creates two circuits as $C_0 = \text{CT-Eval:}1[K^*_{t^*}, \tilde{\alpha^*}, m_b]$ and $C_1 = \text{CT-Eval:}2[K^*_{t^*}, \tilde{\alpha^*}, \tilde{\beta}^*, m_b, m_0]$. In step 8 it submits both of these to the IO challenger and receives back a program P which it passes to the attacker in step 9 as C^*. It executes steps 9-11 as in $\mathsf{Game}\ 6$. If the attacker wins (i.e. $b' = b$), then $\mathcal{B}$ guesses '0' to indicate that P was and obfuscation of C_0; otherwise, it guesses '1' to indicate it was an obfuscation of C_1.

We observe that when P is generated as and obfuscation of C_0, then $\mathcal{B}$ gives exactly the view of $\mathsf{Game}\ 5$ to $\mathcal{A}$. (Note that simply choosing $\tilde{\beta}^*$ and doing nothing with it is equivalent to $\mathsf{Game}\ 5$ from the attacker's view.) Otherwise if P is chosen as an obfuscation of C_1 the view is of $\mathsf{Game}\ 6$. In addition, the programs are functionally equivalent with all but negligible probability. The reason is that $\tilde{\beta}^*$ is outside the image of the pseudo random generator with probability at least $1 - 2^{\lambda}$. And in this case the CT-EVAL:2 program will behave the same as the CT-EVAL:1 program since it only adds a dead branch. Therefore if $\mathsf{Adv}_{\mathcal{A},5} - \mathsf{Adv}_{\mathcal{A},6}$ is non-negligble, $\mathcal{B}$ must also have non-neglgible advantage against the indisguishability obfuscation game. ■

Lemma 11. If our pseudo random generator PRG is secure, then for all PPT $\mathcal{A}$ we have that $\mathsf{Adv}_{\mathcal{A},6} - \mathsf{Adv}_{\mathcal{A},7} = \text{negl}(\lambda)$ for some negligible function negl.

Proof. We describe and analyze a PPT reduction algorithm $\mathcal{B}$ that plays the pseudo random generator security game. $\mathcal{B}$ first receives a PRG game challenge $T \in \{0,1\}^{2\lambda}$. It then runs the attacker and runs the security game as described in Game 6 with the exception that in step 4 it lets $\tilde{\beta}^* = T$. If the attacker wins (i.e. $b' = b$), then $\mathcal{B}$ guesses '1' to indicate that T was in the image of PRG(); otherwise, it outputs '0' to that T was chosen randomly.

We observe that when T is generated as $T = \text{PRG}(r)$, then $\mathcal{B}$ gives exactly the view of Game 6 to $\mathcal{A}$. Otherwise if T is chosen randomly the view is of Game 7. Therefore if $\mathsf{Adv}_{\mathcal{A},6} - \mathsf{Adv}_{\mathcal{A},7}$ is non-negligble, $\mathcal{B}$ must also have non-neglgible advantage against the pseudo random generator. ■

We next observe that Game 7 is identical to Game $8, 0$. We also skip over the lemma dealing with Game $8, i$ to Game $8, I + 1$ and defer it to the next subsection.

Lemma 12. If our pseudo random generator PRG is secure, then for all PPT $\mathcal{A}$ we have that $\mathsf{Adv}_{\mathcal{A},(8,Q(\lambda))} - \mathsf{Adv}_{\mathcal{A},9} = \text{negl}(\lambda)$ for some negligible function negl.

Proof. We describe and analyze a PPT reduction algorithm $\mathcal{B}$ that plays the pseudo random generator security game. $\mathcal{B}$ first receives a PRG game challenge $T \in \{0,1\}^{2\lambda}$. It then runs the attacker and runs the security game as described in Game $8, Q$ with the exception that in step 4 it lets $\tilde{\alpha}^* = T$. We emphasize that in Game $8, Q$ anon of the a_j^* encrypt α. If the attacker wins (i.e. $b' = b$), then $\mathcal{B}$ guesses '1' to indicate that T was in the image of $PRG()$; otherwise, it outputs '0' to that T was chosen randomly.

We observe that when T is generated as $T = \text{PRG}(r)$, then $\mathcal{B}$ gives exactly the view of Game $8, Q$ to $\mathcal{A}$. (We note that step 6b is already the same between these two games.) Otherwise if T is chosen randomly the view is of Game 9. Therefore if $\mathsf{Adv}_{\mathcal{A},8,Q} - \mathsf{Adv}_{\mathcal{A},9}$ is non-negligble, $\mathcal{B}$ must also have non-neglgible advantage against the pseudo random generator. ■

Claim 1. If $i\mathcal{O}$ is a secure indistinguishability obfuscator, then for all PPT $\mathcal{A}$ we have that $\mathsf{Adv}_{\mathcal{A},9} - \mathsf{Adv}_{\mathcal{A},10} = \text{negl}(\lambda)$ for some negligible function negl.

Proof. We describe and analyze a PPT reduction algorithm $\mathcal{B}$ that plays the indistinguishability obfuscation security game with $\mathcal{A}$. $\mathcal{B}$ runs steps 1-7 as in Game 9. Next it creates two circuits as $C_0 =$ CT-EVAL:2$[K_{t^*}^*, \tilde{\alpha^*}, \tilde{\beta}^*, m_b, m_0]$ and and $C_1 =$ CT-EVAL:1$[K_{t^*}^*, \tilde{\beta}^*, m_0]$. It submits both of these to the IO challenger and receives back a program P which it passes to the attacker in step 9 as C^*. It executes steps 9-11 as in Game 9. If the attacker wins (i.e. $b' = b$), then $\mathcal{B}$ guesses '0' to indicate that P was and obfuscation of C_0; otherwise, it guesses '1' to indicate it was an obfuscation of C_1.

We observe that when P is generated as and obfuscation of C_0, then $\mathcal{B}$ gives exactly the view of Game 9 to $\mathcal{A}$. Otherwise if P is chosen as an obfuscation of C_1 the view is of Game 10. In addition, the programs are functionally equivalent with all but negligible probability. The reason is that $\tilde{\alpha}^*$ is outside the image of the pseudo random generator with probability at least $1 - 2^{\lambda}$. And in this case the CT-EVAL:1 program will behave the same as the CT-EVAL:2 program since it only subtracts a dead branch. Therefore if $\mathsf{Adv}_{\mathcal{A},9} - \mathsf{Adv}_{\mathcal{A},10}$ is non-negligble, $\mathcal{B}$ must also have non-neglgible advantage against the indisguishability obfuscation game. ■

6.3 Proving Indistinguishability of $\mathsf{Game}\ 8, i$ and $\mathsf{Game}\ 8, i+1$

Lemma 13. If $i\mathcal{O}$ is a secure indistinguishability obfuscator and our puncturable deterministic encryption scheme is secure, then for all PPT $\mathcal{A}$ and for all $i \in [0, Q(\lambda)-1]$ we have that $\mathsf{Adv}_{\mathcal{A},(8,i)} - \mathsf{Adv}_{\mathcal{A},(8,i+1)} = \text{negl}(\lambda)$ for some negligible function negl.

Proof. The proof of this claim is significantly more complicated than the others and will require the definition of some more hybrid games. We show these as modifications to lines 6-10 of the security game.

$\mathsf{Game}\ 8, i, A$ Same as $\mathsf{Game}\ 8, i$ with the following modifications.

6. Phase 1 Queries: Let f_j be the function of associated with the j-th query.
   (a) Choose random $y_j \in \{0,1\}^\lambda$.
   (b) Compute $(d_j^*, k_j^*, s_{1,j}^*, s_{2,j}^*, s_{3,j}^*) = F_2(K_{t^*}, y_j)$.
   (c) If $j > i$ then set $a_j^* = \mathsf{Encrypt}_{\mathsf{PDE}}(d_j^*, \alpha^*)$; otherwise if $j \le i$ set $a_j^* = \mathsf{Encrypt}_{\mathsf{PDE}}(d_j^*, \beta^*)$.
   (d) Let $\mathrm{SK}_{\mathrm{OB},j}^* = \mathsf{KeyGenOB}(k_j^*, f_j; s_{2,j}^*)$.
   (e) Compute $P_{f_j} \leftarrow i\mathcal{O}(\text{Key-Signal:2}[K(t^*), t^*, a_j^*, \mathrm{SK}_{\mathrm{OB},j}^*, f_j, y_j])$.
   (f) Output the key as (y_j, P_{f_j}).
7. Attacker gives messages $m_0, m_1 \in \mathcal{M}$ to challenger.
8. (a) Choose random $\gamma \in \{0,1\}$. If $\gamma = 0$ let $c_0 = \mathsf{Encrypt}_{\mathsf{PDE}}(d_{i+1}^*, \alpha^*)$, $c_1 = \mathsf{Encrypt}_{\mathsf{PDE}}(d_{i+1}^*, \beta^*)$, $\mathrm{CT}_{\mathrm{OB},0} = \mathsf{EncryptOB}(k_{i+1}^*, m_b; s_{1,i+1}^*)$ and $\mathrm{CT}_{\mathrm{OB},1} = \mathsf{EncryptOB}(k_{i+1}^*, m_0; s_{3,i+1}^*)$.
   Else if $\gamma = 1$ let $c_0 = \mathsf{Encrypt}_{\mathsf{PDE}}(d_{i+1}^*, \beta^*)$ and let $c_1 = \mathsf{Encrypt}_{\mathsf{PDE}}(d_{i+1}^*, \alpha^*)$. $\mathrm{CT}_{\mathrm{OB},0} = \mathsf{EncryptOB}(k_{i+1}^*, m_0; s_{3,i+1}^*)$ and $\mathrm{CT}_{\mathrm{OB},1} = \mathsf{EncryptOB}(k_{i+1}^*, m_b; s_{1,i+1}^*)$.
   (b) Sample $K_{t^*}^*(y_{i+1})$ as $\text{Puncture}_F(K_{t^*}^*, y_{i+1})$.
   (c) Challenger creates $C^* \leftarrow i\mathcal{O}(1^\lambda, \text{CT-Eval:3}[K_{t^*}^*(y_{i+1}), \tilde{\alpha}^*, \tilde{\beta}^*, m_b, m_0, y^*, c_0, c_1, \mathrm{CT}_{\mathrm{OB},0}, \mathrm{CT}_{\mathrm{OB},1}])$.
9. The output ciphertext is $\mathrm{CT} = (t^*, C^*)$.
10. Phase 2 Queries: Same as Phase 1 in step 6.

Remark 3. One (potentially simpler) alternative to randomizing c_0, c_1 by γ is to order c_0, c_1 lexicographically when creating the program CT-Eval:3.

$\mathsf{Game}\ 8, i, B$ Same as $\mathsf{Game}\ 8, i, A$ with the following modifications.

6. Phase 1 Queries: Let f_j be the function of associated with the j-th query.
   (a) Choose random $y_j \in \{0,1\}^\lambda$.
   (b) If $j = i+1$ then choose $(d_j^*, k_j^*, s_{1,j}^*, s_{2,j}^*, s_{3,j}^*)$ uniformly at random.
   Else compute $(d_j^*, k_j^*, s_{1,j}^*, s_{2,j}^*, s_{3,j}^*) = F_2(K_{t^*}, y_j)$.
   (c) If $j > i$ then set $a_j^* = \mathsf{Encrypt}_{\mathsf{PDE}}(d_j^*, \alpha^*)$; otherwise if $j \le i$ set $a_j^* = \mathsf{Encrypt}_{\mathsf{PDE}}(d_j^*, \beta^*)$.
   (d) Let $\mathrm{SK}_{\mathrm{OB},j}^* = \mathsf{KeyGenOB}(k_j^*, f_j; s_{2,j}^*)$.
   (e) Compute $P_{f_j} \leftarrow i\mathcal{O}(\text{Key-Signal:2}[K(t^*), t^*, a_j^*, \mathrm{SK}_{\mathrm{OB},j}^*, f_j, y_j])$.
   (f) Output the key as (y_j, P_{f_j}).
7. Attacker gives messages $m_0, m_1 \in \mathcal{M}$ to challenger.

8. (a) Choose random $\gamma \in \{0,1\}$. If $\gamma = 0$ let $c_0 = \mathsf{Encrypt}_{\mathsf{PDE}}(d^*_{i+1}, \alpha^*)$, $c_1 = \mathsf{Encrypt}_{\mathsf{PDE}}(d^*_{i+1}, \beta^*)$, $\mathrm{CT}_{\mathrm{OB},0} = \mathsf{EncryptOB}(k^*_{i+1}, m_b; s^*_{1,i+1})$ and $\mathrm{CT}_{\mathrm{OB},1} = \mathsf{EncryptOB}(k^*_{i+1}, m_0; s^*_{3,i+1})$.
   Else if $\gamma = 1$ let $c_0 = \mathsf{Encrypt}_{\mathsf{PDE}}(d^*_{i+1}, \beta^*)$ and let $c_1 = \mathsf{Encrypt}_{\mathsf{PDE}}(d^*_{i+1}, \alpha^*)$. $\mathrm{CT}_{\mathrm{OB},0} = \mathsf{EncryptOB}(k^*_{i+1}, m_0; s^*_{3,i+1})$ and $\mathrm{CT}_{\mathrm{OB},1} = \mathsf{EncryptOB}(k^*_{i+1}, m_b; s^*_{1,i+1})$.
   (b) Sample $K^*_{t^*}(y_{i+1})$ as $\mathrm{Puncture}_F(K^*_{t^*}, y_{i+1})$.
   (c) Challenger creates $C^* \leftarrow i\mathcal{O}(1^\lambda, \text{CT-EVAL:3}[K^*_{t^*}(y_{i+1}), \tilde{\alpha^*}, \tilde{\beta^*}, m_b, m_0, y^*, c_0, c_1, \mathrm{CT}_{\mathrm{OB},0}, \mathrm{CT}_{\mathrm{OB},1}])$.
9. The output ciphertext is $\mathrm{CT} = (t^*, C^*)$.
10. Phase 2 Queries: Same as Phase 1 in step 6.

$\mathsf{Game}\ 8, i, C$ Same as $\mathsf{Game}\ 8, i, B$ with the following modifications.

6. Phase 1 Queries: Let f_j be the function of associated with the j-th query.
   (a) Choose random $y_j \in \{0,1\}^\lambda$.
   (b) If $j = i+1$ then choose $(d^*_j, k^*_j, s^*_{1,j}, s^*_{2,j}, s^*_{3,j})$ uniformly at random.
   Else compute $(d^*_j, k^*_j, s^*_{1,j}, s^*_{2,j}, s^*_{3,j}) = F_2(K_{t^*}, y_j)$.
   (c) If $j > i$ then set $a^*_j = \mathsf{Encrypt}_{\mathsf{PDE}}(d^*_j, \alpha^*)$; otherwise if $j \le i$ set $a^*_j = \mathsf{Encrypt}_{\mathsf{PDE}}(d^*_j, \beta^*)$.
   (d) Let $\mathrm{SK}^*_{\mathrm{OB},j} = \mathsf{KeyGenOB}(k^*_j, f_j; s^*_{2,j})$.
   (e) Compute $P_{f_j} \leftarrow i\mathcal{O}(\text{KEY-SIGNAL:2}[K(t^*), t^*, a^*_j, \mathrm{SK}^*_{\mathrm{OB},j}, f_j, y_j])$.
   (f) Output the key as (y_j, P_{f_j}).
7. Attacker gives messages $m_0, m_1 \in \mathcal{M}$ to challenger.
8. (a) Choose random $\gamma \in \{0,1\}$. If $\gamma = 0$ let $c_0 = \mathsf{Encrypt}_{\mathsf{PDE}}(d^*_{i+1}, \alpha^*)$, $c_1 = \mathsf{Encrypt}_{\mathsf{PDE}}(d^*_{i+1}, \beta^*)$, $\mathrm{CT}_{\mathrm{OB},0} = \underline{\mathsf{EncryptOB}(k^*_{i+1}, m_0; s^*_{1,i+1})}$ and $\mathrm{CT}_{\mathrm{OB},1} = \mathsf{EncryptOB}(k^*_{i+1}, m_0; s^*_{3,i+1})$.
   Else if $\gamma = 1$ let $c_0 = \mathsf{Encrypt}_{\mathsf{PDE}}(d^*_{i+1}, \beta^*)$ and let $c_1 = \mathsf{Encrypt}_{\mathsf{PDE}}(d^*_{i+1}, \alpha^*)$. $\mathrm{CT}_{\mathrm{OB},0} = \mathsf{EncryptOB}(k^*_{i+1}, m_0; s^*_{3,i+1})$ and $\mathrm{CT}_{\mathrm{OB},1} = \underline{\mathsf{EncryptOB}(k^*_{i+1}, m_0; s^*_{1,i+1})}$.
   (b) Sample $K^*_{t^*}(y_{i+1})$ as $\mathrm{Puncture}_F(K^*_{t^*}, y_{i+1})$.
   (c) Challenger creates $C^* \leftarrow i\mathcal{O}(1^\lambda, \text{CT-EVAL:3}[K^*_{t^*}(y_{i+1}), \tilde{\alpha^*}, \tilde{\beta^*}, m_b, m_0, y^*, c_0, c_1, \mathrm{CT}_{\mathrm{OB},0}, \mathrm{CT}_{\mathrm{OB},1}])$.
9. The output ciphertext is $\mathrm{CT} = (t^*, C^*)$.
10. Phase 2 Queries: Same as Phase 1 in step 6.

$\mathsf{Game}\ 8, i, D$ Same as $\mathsf{Game}\ 8, i, C$ with the following modifications.

6. Phase 1 Queries: Let f_j be the function of associated with the j-th query.
   (a) Choose random $y_j \in \{0,1\}^\lambda$.
   (b) If $j = i+1$ then choose $(d^*_j, k^*_j, s^*_{1,j}, s^*_{2,j}, s^*_{3,j})$ uniformly at random.
   Else compute $(d^*_j, k^*_j, s^*_{1,j}, s^*_{2,j}, s^*_{3,j}) = F_2(K_{t^*}, y_j)$.
   (c) $\underline{\text{If } j > i+1}$ then set $a^*_j = \mathsf{Encrypt}_{\mathsf{PDE}}(d^*_j, \alpha^*)$; otherwise if $\underline{j \le i+1}$ set $a^*_j = \mathsf{Encrypt}_{\mathsf{PDE}}(d^*_j, \beta^*)$.
   (d) Let $\mathrm{SK}^*_{\mathrm{OB},j} = \mathsf{KeyGenOB}(k^*_j, f_j; s^*_{2,j})$.
   (e) Compute $P_{f_j} \leftarrow i\mathcal{O}(\text{KEY-SIGNAL:2}[K(t^*), t^*, a^*_j, \mathrm{SK}^*_{\mathrm{OB},j}, f_j, y_j])$.
   (f) Output the key as (y_j, P_{f_j}).
7. Attacker gives messages $m_0, m_1 \in \mathcal{M}$ to challenger.

8. (a) Choose random $\gamma \in \{0,1\}$. If $\gamma = 0$ let $c_0 = \mathsf{Encrypt}_{\mathsf{PDE}}(d^*_{i+1}, \alpha^*)$, $c_1 = \mathsf{Encrypt}_{\mathsf{PDE}}(d^*_{i+1}, \beta^*)$, $\mathrm{CT}_{\mathrm{OB},0} = \mathsf{EncryptOB}(k^*_{i+1}, m_0; s^*_{1,i+1})$ and $\mathrm{CT}_{\mathrm{OB},1} = \mathsf{EncryptOB}(k^*_{i+1}, m_0; s^*_{3,i+1})$.
Else if $\gamma = 1$ let $c_0 = \mathsf{Encrypt}_{\mathsf{PDE}}(d^*_{i+1}, \beta^*)$ and let $c_1 = \mathsf{Encrypt}_{\mathsf{PDE}}(d^*_{i+1}, \alpha^*)$. $\mathrm{CT}_{\mathrm{OB},0} = \mathsf{EncryptOB}(k^*_{i+1}, m_0; s^*_{3,i+1})$ and $\mathrm{CT}_{\mathrm{OB},1} = \mathsf{EncryptOB}(k^*_{i+1}, m_0; s^*_{1,i+1})$.
   (b) Sample $K^*_{t^*}(y_{i+1})$ as $\mathrm{Puncture}_F(K^*_{t^*}, y_{i+1})$.
   (c) Challenger creates $C^* \leftarrow i\mathcal{O}(1^\lambda, \text{CT-EVAL:3}[K^*_{t^*}(y_{i+1}), \tilde{\alpha^*}, \tilde{\beta^*}, m_b, m_0, y^*, c_0, c_1, \mathrm{CT}_{\mathrm{OB},0}, \mathrm{CT}_{\mathrm{OB},1}])$.
9. The output ciphertext is $\mathrm{CT} = (t^*, C^*)$.
10. Phase 2 Queries: Same as Phase 1 in step 6.

Now that the signal has been changed we reverse out of the modifications we have been making.

Game $8, i, E$ Same as **Game** $8, i, D$ with the following modifications.

6. Phase 1 Queries: Let f_j be the function of associated with the j-th query.
   (a) Choose random $y_j \in \{0,1\}^\lambda$.
   (b) If $j = i+1$ then choose $(d^*_j, k^*_j, s^*_{1,j}, s^*_{2,j}, s^*_{3,j})$ uniformly at random.
Else compute $(d^*_j, k^*_j, s^*_{1,j}, s^*_{2,j}, s^*_{3,j}) = F_2(K_{t^*}, y_j)$.
   (c) If $j > i+1$ then set $a^*_j = \mathsf{Encrypt}_{\mathsf{PDE}}(d^*_j, \alpha^*)$; otherwise if $j \le i+1$ set $a^*_j = \mathsf{Encrypt}_{\mathsf{PDE}}(d^*_j, \beta^*)$.
   (d) Let $\mathrm{SK}^*_{\mathrm{OB},j} = \mathsf{KeyGenOB}(k^*_j, f_j; s^*_{2,j})$.
   (e) Compute $P_{f_j} \leftarrow i\mathcal{O}(\text{KEY-SIGNAL:2}[K(t^*), t^*, a^*_j, \mathrm{SK}^*_{\mathrm{OB},j}, f_j, y_j])$.
   (f) Output the key as (y_j, P_{f_j}).
7. Attacker gives messages $m_0, m_1 \in \mathcal{M}$ to challenger.
8. (a) Choose random $\gamma \in \{0,1\}$. If $\gamma = 0$ let $c_0 = \mathsf{Encrypt}_{\mathsf{PDE}}(d^*_{i+1}, \alpha^*)$, $c_1 = \mathsf{Encrypt}_{\mathsf{PDE}}(d^*_{i+1}, \beta^*)$, $\mathrm{CT}_{\mathrm{OB},0} = \mathsf{EncryptOB}(k^*_{i+1}, m_b; s^*_{1,i+1})$ and $\mathrm{CT}_{\mathrm{OB},1} = \mathsf{EncryptOB}(k^*_{i+1}, m_0; s^*_{3,i+1})$.
Else if $\gamma = 1$ let $c_0 = \mathsf{Encrypt}_{\mathsf{PDE}}(d^*_{i+1}, \beta^*)$ and let $c_1 = \mathsf{Encrypt}_{\mathsf{PDE}}(d^*_{i+1}, \alpha^*)$. $\mathrm{CT}_{\mathrm{OB},0} = \mathsf{EncryptOB}(k^*_{i+1}, m_0; s^*_{3,i+1})$ and $\mathrm{CT}_{\mathrm{OB},1} = \mathsf{EncryptOB}(k^*_{i+1}, m_b; s^*_{1,i+1})$.
   (b) Sample $K^*_{t^*}(y_{i+1})$ as $\mathrm{Puncture}_F(K^*_{t^*}, y_{i+1})$.
   (c) Challenger creates $C^* \leftarrow i\mathcal{O}(1^\lambda, \text{CT-EVAL:3}[K^*_{t^*}(y_{i+1}), \tilde{\alpha^*}, \tilde{\beta^*}, m_b, m_0, y^*, c_0, c_1, \mathrm{CT}_{\mathrm{OB},0}, \mathrm{CT}_{\mathrm{OB},1}])$.
9. The output ciphertext is $\mathrm{CT} = (t^*, C^*)$.
10. Phase 2 Queries: Same as Phase 1 in step 6.

Game $8, i, F$ Same as **Game** $8, i, E$ with the following modifications.

6. Phase 1 Queries: Let f_j be the function of associated with the j-th query.
   (a) Choose random $y_j \in \{0,1\}^\lambda$.
   (b) Compute $(d^*_j, k^*_j, s^*_{1,j}, s^*_{2,j}, s^*_{3,j}) = F_2(K_{t^*}, y_j)$.
   (c) If $j > i+1$ then set $a^*_j = \mathsf{Encrypt}_{\mathsf{PDE}}(d^*_j, \alpha^*)$; otherwise if $j \le i+1$ set $a^*_j = \mathsf{Encrypt}_{\mathsf{PDE}}(d^*_j, \beta^*)$.
   (d) Let $\mathrm{SK}^*_{\mathrm{OB},j} = \mathsf{KeyGenOB}(k^*_j, f_j; s^*_{2,j})$.
   (e) Compute $P_{f_j} \leftarrow i\mathcal{O}(\text{KEY-SIGNAL:2}[K(t^*), t^*, a^*_j, \mathrm{SK}^*_{\mathrm{OB},j}, f_j, y_j])$.
   (f) Output the key as (y_j, P_{f_j}).
7. Attacker gives messages $m_0, m_1 \in \mathcal{M}$ to challenger.

8. (a) Choose random $\gamma \in \{0,1\}$. If $\gamma = 0$ let $c_0 = \mathsf{Encrypt}_{\mathsf{PDE}}(d^*_{i+1}, \alpha^*)$, $c_1 = \mathsf{Encrypt}_{\mathsf{PDE}}(d^*_{i+1}, \beta^*)$, $\mathrm{CT}_{\mathrm{OB},0} = \mathsf{EncryptOB}(k^*_{i+1}, m_b; s^*_{1,i+1})$ and $\mathrm{CT}_{\mathrm{OB},1} = \mathsf{EncryptOB}(k^*_{i+1}, m_0; s^*_{3,i+1})$.
Else if $\gamma = 1$ let $c_0 = \mathsf{Encrypt}_{\mathsf{PDE}}(d^*_{i+1}, \beta^*)$ and let $c_1 = \mathsf{Encrypt}_{\mathsf{PDE}}(d^*_{i+1}, \alpha^*)$. $\mathrm{CT}_{\mathrm{OB},0} = \mathsf{EncryptOB}(k^*_{i+1}, m_0; s^*_{3,i+1})$ and $\mathrm{CT}_{\mathrm{OB},1} = \mathsf{EncryptOB}(k^*_{i+1}, m_b; s^*_{1,i+1})$.
   (b) Sample $K^*_{t^*}(y_{i+1})$ as $\mathrm{Puncture}_F(K^*_{t^*}, y_{i+1})$.
   (c) Challenger creates $C^* \leftarrow i\mathcal{O}(1^\lambda, \text{CT-EVAL:3}[K^*_{t^*}(y_{i+1}), \tilde{\alpha^*}, \tilde{\beta^*}, m_b, m_0, y^*, c_0, c_1, \mathrm{CT}_{\mathrm{OB},0}, \mathrm{CT}_{\mathrm{OB},1}])$.
9. The output ciphertext is $\mathrm{CT} = (t^*, C^*)$.
10. Phase 2 Queries: Same as Phase 1 in step 6.

$\mathsf{Game}\ 8, i, G$ Same as $\mathsf{Game}\ 8, i, F$ with the following modifications.

6. Phase 1 Queries: Let f_j be the function of associated with the j-th query.
   (a) Choose random $y_j \in \{0,1\}^\lambda$.
   (b) Compute $(d^*_j, k^*_j, s^*_{1,j}, s^*_{2,j}, s^*_{3,j}) = F_2(K_{t^*}, y_j)$.
   (c) If $j > i+1$ then set $a^*_j = \mathsf{Encrypt}_{\mathsf{PDE}}(d^*_j, \alpha^*)$; otherwise if $j \le i+1$ set $a^*_j = \mathsf{Encrypt}_{\mathsf{PDE}}(d^*_j, \beta^*)$.
   (d) Let $\mathrm{SK}^*_{\mathrm{OB},j} = \mathsf{KeyGenOB}(k^*_j, f_j; s^*_{2,j})$.
   (e) Compute $P_{f_j} \leftarrow i\mathcal{O}(\text{KEY-SIGNAL:2}[K(t^*), t^*, a^*_j, \mathrm{SK}^*_{\mathrm{OB},j}, f_j, y_j])$.
   (f) Output the key as (y_j, P_{f_j}).
7. Attacker gives messages $m_0, m_1 \in \mathcal{M}$ to challenger.
8. Challenger sets the program $C^* \leftarrow i\mathcal{O}(1^\lambda, \text{CT-EVAL:2}[K^*_{t^*}, \tilde{\alpha^*}, \tilde{\beta^*}, m_b, m_0])$.
9. The output ciphertext is $\mathrm{CT} = (t^*, C^*)$.
10. Phase 2 Queries: Same as Phase 1 in step 6.

We conclude by observing that $\mathsf{Game}\ 8, i, G$ is identical to $\mathsf{Game}\ 8, i+1$. We now give our indistinguishability claims.

CT-Eval:3

Constants: PRF key $K_t(y^*)$, $\tilde{\alpha}, \tilde{\beta} \in \{0,1\}^{2\cdot\lambda}$, messages $m, m_{\mathrm{fixed}} \in \mathcal{M}$, y^*, c_0, c_1, $\mathrm{CT}_{\mathrm{OB},0}$, $\mathrm{CT}_{\mathrm{OB},1}$.
Input: PDE ciphertext a and $y \in \{0,1\}^\lambda$.

1. If $y = y^*$ AND $(a = c_0)$ output $\mathrm{CT}_{\mathrm{OB},0}$.
2. If $y = y^*$ AND $(a = c_1)$ output $\mathrm{CT}_{\mathrm{OB},1}$.
3. If $y = y^*$ AND $(a \neq c_0, c_1)$ then output a rejecting $\perp$.
4. Else if $y \neq y^*$ Compute $(d, k, s_1, s_2, s_3) = F_2(K_t(y^*), y)$.
5. Compute $e = \mathsf{Decrypt}_{\mathsf{PDE}}(d, a)$.
6. If $\mathrm{PRG}(e) = \tilde{\alpha}$ output $\mathsf{EncryptOB}(k, m; s_1)$.
7. If $\mathrm{PRG}(e) = \tilde{\beta}$ output $\mathsf{EncryptOB}(k, m; s_3)$.
8. Else output a rejecting $\perp$.

Figure 11: Program CT-Eval:3

■

Game $8, i$ **to** Game $8, i, A$

Claim 2. If $i\mathcal{O}$ is a secure indistinguishability obfuscator, then for all PPT $\mathcal{A}$ we have that $\mathsf{Adv}_{\mathcal{A},8,i} - \mathsf{Adv}_{\mathcal{A},8,i,A} = \text{negl}(\lambda)$ for some negligible function negl.

We describe and analyze a PPT reduction algorithm $\mathcal{B}$ that plays the indistinguishability obfuscation security game with $\mathcal{A}$. $\mathcal{B}$ runs steps 1-7 as in Game $8, i, A$. Next it creates two circuits as $C_0 = \text{CT-EVAL:}2[K^*_{t^*}, \tilde{\alpha^*}, \tilde{\beta^*}, m_b, m_0]$ and and $C_1 = \text{CT-EVAL:}3[K^*_{t^*}(y_{i+1}), \tilde{\alpha^*}, \tilde{\beta^*}, m_b, m_0, y^*, c_0, c_1, \text{CT}_{\text{OB},0}, \text{CT}_{\text{OB},1}]$. It submits both of these to the IO challenger and receives back a program P which it passes to the attacker in step 9 as C^*. It turns steps 9-11 as in Game $8, i, A$. If the attacker wins (i.e. $b' = b$), then $\mathcal{B}$ guesses '0' to indicate that P was and obfuscation of C_0; otherwise, it guesses '1' to indicate it was an obfuscation of C_1.

We observe that when P is generated as and obfuscation of C_0, then $\mathcal{B}$ gives exactly the view of Game $8, i$ to $\mathcal{A}$. Otherwise if P is chosen as an obfuscation of C_1 the view is of Game $8, i, A$. In addition, the programs are functionally equivalent. The reason is that the programs have the same behavior except for the difference that the CT-EVAL:3 program has the CT-EVAL:2 program's behavior hardwired in a two points and uses a punctured key at another place. *The hardwiring is only needed for two points since the PDE system is deterministic and the PRG is injective.* Therefore if $\mathsf{Adv}_{\mathcal{A},8,i} - \mathsf{Adv}_{\mathcal{A},8,i,A}$ is non-negligble, $\mathcal{B}$ must also have non-neglgible advantage against the indisguishability obfuscation game.

Game $8, i, A$ **to** Game $8, i, B$

Claim 3. If F is a selectively secure puncturable PRF then for all PPT $\mathcal{A}$ we have that $\mathsf{Adv}_{\mathcal{A},8,i,A} - \mathsf{Adv}_{\mathcal{A},8,i,B} = \text{negl}(\lambda)$ for some negligible function negl.

Proof. We describe and analyze a PPT reduction algorithm $\mathcal{B}$ that plays the selective puncturable PRF security game. $\mathcal{B}$ begins by choosing $y_{i+1} \in \{0,1\}^\lambda$ at step 3. It then chooses submits this to the punctured PRF challenger for function F_2. It receives back a punctured key $K^*_{t^*}(y_{i+1})$ and a challenge value $z \in \{0,1\}^{5\lambda}$. It runs steps 2 onward for $\mathcal{A}$ as in Game $8, i$. When making private key $i+1$ (in either Phase 1 or 2) the challenger sets $(d^*_j, k^*_j, s^*_{1,j}, s^*_{2,j}, s^*_{3,j}) = z$. All other steps are simulated by the reduction with the exception that we abort if $y_j = y_{i+1}$ for $j \neq i+1$. This abort condition occurs with negligible probability so we can ignore it. We emphasize that in step 8, the CT-EVAL:3 program is parameterized by the punctured key $K^*_{t^*}(y_{i+1})$. If the attacker wins (i.e. $b' = b$), then $\mathcal{B}$ guesses '1' to indicated that $z = F(K, t^*)$; otherwise, it outputs '0' to that z was chosen randomly.

We observe that when z is generated as $F(K, t^*)$, then $\mathcal{B}$ gives exactly the view of Game $8, i, A$ to $\mathcal{A}$. Otherwise if z is chosen randomly the view is of Game $8, i, B$. Therefore if $\mathsf{Adv}_{\mathcal{A},8,i,A} - \mathsf{Adv}_{\mathcal{A},8,i,B}$ is non-negligble, $\mathcal{B}$ must also have non-neglgible advantage against the security of the puncturable PRF.

■

Game $8, i, B$ **to** Game $8, i, C$

Claim 4. If (KeyGenOB, EncryptOB, DecryptOB) is an adaptively secure 1-bounded functional encryption system with master key encryption, then then for all PPT $\mathcal{A}$ we have that $\mathsf{Adv}_{\mathcal{A},8,i,B} - \mathsf{Adv}_{\mathcal{A},8,i,C} = \text{negl}(\lambda)$ for some negligible function negl.

Proof. We describe and analyze a PPT reduction algorithm $\mathcal{B}$ that plays the 1-bounded functional encryption security game. We begin by noting that an attacker can only have a non-zero difference in advantage between the two games when the bit $b = 1$. Otherwise, they appear identical. So we condition the reduction on setting the bit $b = 1$.

Suppose that the $i+1$-th query is in Phase 1. The reduction algorithm runs the experiment through step 5. For step 6 it creates all secret keys itself except for the $i+1$ key. For this the reduction algorithm queries the FE challenger with f_{i+1} and receives back $\text{SK}^*_{\text{OB},i+1}$.

We now move to step in creating the challenge ciphertext. The reduction algorithm flips a coin $\gamma \in \{0,1\}$ as in step 8a. It then queries the encryption oracle for an encryption of m_0. It sets the reply (which is not the one-bounded challenge ciphertext) to be $\mathrm{CT}_{\mathrm{OB},1-\gamma}$. Next it submits (m_0, m_1) to the FE challenger and receives back the one bounded challenge ciphertext. It sets this to be be $\mathrm{CT}_{\mathrm{OB},\gamma}$. The values c_0, c_1 are set according to γ as in step 8a. The reduction algorithm then runs the rest of the experiment itself.

If the attacker wins (i.e. $b' = b$), then $\mathcal{B}$ guesses '1' to indicated that the challenge ciphertext was an encryption of m_1; otherwise, it outputs '0' to indicate that m_0 was encrypted.

In the case that the $i+1$-th query was in Phase 2, the reduction is the same except that the challenge ciphertext is queried before the key. Either order is okay since the 1-bounded scheme is assumed to be adaptively secure. We emphasize that our reduction only makes a single key query.

We observe that when the challenge ciphertext encrypts m_1, then $\mathcal{B}$ gives exactly the view of $\mathsf{Game}\ 8, i, B$ to $\mathcal{A}$. Otherwise if m_0 were encrypted, then the view is of $\mathsf{Game}\ 8, i, C$. Therefore if $\mathsf{Adv}_{\mathcal{A},8,i,B} - \mathsf{Adv}_{\mathcal{A},8,i,C}$ is non-negligble, $\mathcal{B}$ must also have non-neglgible advantage against the security of the 1-bounded FE scheme.

■

$\mathsf{Game}\ 8, i, C$ **to** $\mathsf{Game}\ 8, i, D$

Claim 5. If our puncturable deterministic encryption scheme is secure then for all PPT $\mathcal{A}$ we have that $\mathsf{Adv}_{\mathcal{A},8,C} - \mathsf{Adv}_{\mathcal{A},8,D} = \mathrm{negl}(\lambda)$ for some negligible function negl.

Proof. We describe and analyze a PPT reduction algorithm $\mathcal{B}$ that plays the pseudo random generator security game. Suppose that query $i+1$ is in Phase 1.

$\mathcal{B}$ first executes steps 1-5 as in $\mathsf{Game}\ 8, C$ as well as answer key queries j for all $j \neq i+1$. Then it submits messages (α^*, β^*) to the PDE challenger and receives back (T_0, T_1). In step 6 it sets $a^*_{i+1} = T_0$. It then runs until step 8, where it chooses random γ and sets $c_0 = T_\gamma$ and $c_1 = T_{1-\gamma}$. The reduction algorithm simulates the rest of the game. If the attacker wins (i.e. $b' = b$), then $\mathcal{B}$ guesses '0' to indicated that c^* was an encryption of α^*; otherwise, it outputs '1' to that c^* was an encryption of β^*.

We make two important observations. The first is that it is in this security proof where choosing a random γ (i.e. randomizing the assignment of c_0 and c_1 is important. In addition, the security proof also works because at this point both of the hardwired one-bounded encryptions are of the same message m_0.

When T_0 is generated as $\mathsf{Encrypt}_{\mathsf{PDE}}(k^*, \alpha^*)$ then $\mathcal{B}$ gives exactly the view of $\mathsf{Game}\ 8, C$. Otherwise when T_0 is generated as $\mathsf{Encrypt}_{\mathsf{PDE}}(k^*, \beta^*)$ the view is of $\mathsf{Game}\ 8, D$. Therefore if $\mathsf{Adv}_{\mathcal{A},8,C} - \mathsf{Adv}_{\mathcal{A},8,D}$ is non-negligble, $\mathcal{B}$ must also have non-neglgible advantage against the puncturable deterministic encryption system.

■

$\mathsf{Game}\ 8, i, D$ **to** $\mathsf{Game}\ 8, i, E$

Claim 6. If $(\mathsf{KeyGenOB}, \mathsf{EncryptOB}, \mathsf{DecryptOB})$ is an adaptively secure 1-bounded functional encryption system with master key encryption, then then for all PPT $\mathcal{A}$ we have that $\mathsf{Adv}_{\mathcal{A},8,i,D} - \mathsf{Adv}_{\mathcal{A},8,i,E} = \mathrm{negl}(\lambda)$ for some negligible function negl.

The proof of this claim is analogous to that of Claim 4.

$\mathsf{Game}\ 8, i, E$ **to** $\mathsf{Game}\ 8, i, F$

Claim 7. If F is a selectively secure puncturable PRF then for all PPT $\mathcal{A}$ we have that $\mathsf{Adv}_{\mathcal{A},8,i,E} - \mathsf{Adv}_{\mathcal{A},8,i,F} = \mathrm{negl}(\lambda)$ for some negligible function negl.

The proof of this claim is analogous to that of Claim 3.

$\mathsf{Game}\ 8, i, F$ **to** $\mathsf{Game}\ 8, i, G$

Claim 8. If $i\mathcal{O}$ is a secure indistinguishability obfuscator, then for all PPT $\mathcal{A}$ we have that $\mathsf{Adv}_{\mathcal{A},8,i,F} - \mathsf{Adv}_{\mathcal{A},8,i,G} = \text{negl}(\lambda)$ for some negligible function negl.

The proof of this claim is analogous to that of Claim 2.

To wrap things up we observe that Lemma 13 follows from a hybrid argument using the claims above. Finally, our main security Theorem 4 follows via hybrid argument from the established lemmas.

7 Using Non-Injective Pseudo Random Generators

Our adaptive construction required the use of an injective pseudo random generator. In this section we informally sketch how to modify our construction to handle non-injective PRGs. We describe the modification in two main steps.

We first observe that if PRG is non-injective then there could be multiple pre-images to $\tilde{\alpha}^*$ and $\tilde{\beta}^*$ in addition to α^* and β^*. Therefore we would need to adjust program CT-Eval:3 so that when $y = y^*$ AND $(a \neq c_0, c_1)$ it attempts to decrypt a as opposed to simply outputting $\perp$. This can be securely accomplished by giving the program the punctured PDE key $d(\alpha^*, \beta^*)$, which can be used to decrypt all ciphertexts except c_0 and c_1.

The above modification will allow all lemmas and claims of the existing proof to go through *except* for the claim of indistinguishability of $\mathsf{Game}\ 8, i, C$ and $\mathsf{Game}\ 8, i, D$. The problem with this game is that all "$\tilde{\alpha}^*$ ciphertexts" are the same since they are encrypted with the same randomness $s^*_{1,i+1}$. A similar problem occurs with the "$\tilde{\beta}^*$ ciphertexts".

A solution is to modify the scheme such that the randomness for creating the one-bounded ciphertexts does not come directly out of F_2 in the CT-Eval programs. Instead, F_2 could output a master key of yet another puncturable PRF F_3. This puncturable PRF would then take in a PDE ciphertext and output the randomness used for a creating a one-bounded ciphertext. The proof would need to be adjusted with an additional puncturable PRF step.

We emphasize that the above argument is informal intuition why we believe the system can be adjusted to handle non-injective PRGs and we do not make any formal claims of such. We choose to pursue using injective PRGs in our formal construction and proof to help avoid additional complexity in our exposition.

References

[ABG+13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689, 2013. http://eprint.iacr.org/.

[BCOP04] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *EUROCRYPT*, pages 506–522, 2004.

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In *TCC*, pages 52–73, 2014.

[BFO08] Alexandra Boldyreva, Serge Fehr, and Adam O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In *CRYPTO*, pages 335–359, 2008.

[BFOR08] Mihir Bellare, Marc Fischlin, Adam O'Neill, and Thomas Ristenpart. Deterministic encryption: Definitional equivalences and constructions without random oracles. In *CRYPTO*, pages 360–378, 2008.

[BGI$^+$12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6, 2012.

[BGI13] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. *IACR Cryptology ePrint Archive*, 2013:401, 2013.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: definitions and challenges. In *Proceedings of the 8th conference on Theory of cryptography*, TCC'11, pages 253–273, Berlin, Heidelberg, 2011. Springer-Verlag.

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *Proceedings of the 4th conference on Theory of cryptography*, TCC'07, pages 535–554, Berlin, Heidelberg, 2007. Springer-Verlag.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. *IACR Cryptology ePrint Archive*, 2013:352, 2013.

[CJO$^+$13] Angelo De Caro, Vincenzo Iovino Abhishek Jain, Adam O'Neill, Omer Paneth, and Giuseppe Persiano. On the achievability of simulation-based security for functional encryption. In *CRYPTO*, 2013.

[CW14] Jie Chen and Hoeteck Wee. Semi-adaptive attribute-based encryption and improved delegation for boolean formula. SCN (To appear), 2014. `http://eprint.iacr.org/`.

[GGH$^+$13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013.

[GGHW14] Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In *CRYPTO*, pages 518–535, 2014.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *FOCS*, pages 464–479, 1984.

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In *CRYPTO*, 2012.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. *IACR Cryptology ePrint Archive*, 2013:379, 2013.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Proceedings of the theory and applications of cryptographic techniques 27th annual international conference on Advances in cryptology*, EUROCRYPT'08, 2008.

[KSW14] Dakshita Khurana, Amit Sahai, and Brent Waters. How to generate and use universal parameters. Cryptology ePrint Archive, Report 2014/507, 2014. `http://eprint.iacr.org/`.

[LOS$^+$10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT*, pages 62–91, 2010.

[OT10] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *CRYPTO*, pages 191–208, 2010.

[SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In *Proceedings of the 17th ACM conference on Computer and communications security*, CCS '10, pages 463–472, New York, NY, USA, 2010. ACM.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.

[SW08] Amit Sahai and Brent Waters. Slides on functional encryption. PowerPoint presentation, 2008. `http://www.cs.utexas.edu/~bwaters/presentations/files/functional.ppt`.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *STOC*, pages 475–484, 2014.

[Wat09] Brent Waters. Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In *CRYPTO*, pages 619–636, 2009.