Semi-Adaptive Security and Bundling Functionalities Made Generic and Easy

Rishab Goyal
rgoyal@cs.utexas.edu

Venkata Koppula
kvenkata@cs.utexas.edu

Brent Waters
bwaters@cs.utexas.edu*

Abstract

Semi-adaptive security is a notion of security that lies between selective and adaptive security for Attribute-Based Encryption (ABE) and Functional Encryption (FE) systems. In the semi-adaptive model the attacker is forced to disclose the challenge messages before it makes any key queries, but is allowed to see the public parameters.

We show how to generically transform any selectively secure ABE or FE scheme into one that is semi-adaptively secure with the only additional assumption being public key encryption, which is already naturally included in almost any scheme of interest. Our technique utilizes a fairly simple application of garbled circuits where instead of encrypting directly, the encryptor creates a garbled circuit that takes as input the public parameters and outputs a ciphertext in the underlying selective scheme. Essentially, the encryption algorithm encrypts without knowing the 'real' public parameters. This allows one to delay giving out the underlying selective parameters until a private key is issued, which connects the semi-adaptive to selective security. The methods used to achieve this result suggest that the moral gap between selective and semi-adaptive security is in general much smaller than that between semi-adaptive and full security.

Finally, we show how to extend the above idea to generically bundle a family of functionalities under one set of public parameters. For example, suppose we had an inner product predicate encryption scheme where the length of the vectors was specified at setup and therefore fixed to the public parameters. Using our transformation one could create a system where for a single set of public parameters the vector length is not apriori bounded, but instead is specified by the encryption algorithm. The resulting ciphertext would be compatible with any private key generated to work on the same input length.

1 Introduction

Traditionally, in a public key encryption system a user will encrypt data m under a second user's public key to create a ciphertext. A receiver of the ciphertext can decrypt the data if they possess the corresponding secret key; otherwise, they will learn nothing. Over the last several years there has been a dramatic re-envisioning of the expressiveness of encryption systems with the introduction of Identity-Based Encryption (IBE) [34, 12, 19], Attribute-Based Encryption (ABE) [32] and culminating in Functional Encryption (FE) [33], which encompasses IBE and ABE.

In these systems a setup algorithm produces a master public/secret key pair, where the master public key is made public and the master secret key is retained by an authority. Any user can encrypt data m using the public parameters[1] to produce a ciphertext ct. In parallel the authority may issue (any number of times) to a user a secret key sk_f that allows the user to learn the output $f(m)$ of a ciphertext that encrypts data m. The message space $\mathcal{M}$ and function space $\mathcal{F}$ allowed depend on the expressiveness of the underlying cryptosystem.

*Supported by NSF CNS-1228599 and CNS-1414082, DARPA SafeWare, Microsoft Faculty Fellowship, and Packard Foundation Fellowship.

[1] We use public parameters and master public key interchangeably.

The security of this class of systems is captured by an indistinguishability based security game between a challenger and an attacker.[2] In this game the challenger will first generate the master public key that it sends to the attacker. The attacker begins by entering the first key query phase where it will issue a polynomial number of key queries, each for a functionality $f \in \mathcal{F}$. For each query the attacker receives back a corresponding secret key SK_f. Next the attacker submits two challenge messages m_0, m_1 with the restriction that $f(m_0) = f(m_1)$ for all functions f queried on earlier. The challenger will flip a coin $b \in \{0, 1\}$ and return a challenge ciphertext ct^* encrypting m_b. Next, the attacker will engage in a second set of private key queries with the same restrictions. Finally, it will output a guess $b' \in \{0, 1\}$ and win if $b = b'$. For any secure scheme the probability of winning should be negligibly close to $\frac{1}{2}$.

The above game, called *full* or *adaptive* security game, captures our intuitive notion of what an indistinguishability based security game should look like. Namely, that an attacker cannot distinguish between two messages unless he receives keys that trivially allow him to — even if the attacker gets to adaptively choose what the keys and messages are. One issue faced by researchers is that when striving for a new functionality it is often difficult at first to achieve full security if we want to restrict ourselves to polynomial loss in the reductions and avoid relying on sub-exponential hardness assumptions. To ease the initial pathway people often consider security under a weaker notion of *selective* [17] security where the attacker is forced to submit the challenge messages m_0, m_1 *before* seeing the public parameters. After gaining this foothold, later work can circle back to move from selective to adaptive security.

Over the past decade there have been several examples of this process in achieving adaptive security for IBE, ABE and FE. The first such examples were the "partitioning" techniques developed by Boneh and Boyen [11] and Waters [35] in the context of achieving Identity-Based Encryption in the standard model, improving upon earlier selectively secure realizations [17, 10]. While partitioning methods were helpful in realizing full security for IBE, they did not generalize to more complex functionalities. To that end a new set of techniques were developed to move beyond partitioning which include those by Gentry and Halevi [21, 22] and Waters' Dual System Encryption [36] methodology. The latter which spawned several other works within that methodology, e.g., [27, 29, 39].

More recently, Ananth et al. [2], building upon the bootstrapping concepts of [38], showed how to generically convert an FE scheme that supports arbitrary poly-sized circuits from selective security into one that achieves full security.[3] Their result, however, does not apply to the many ABE or FE schemes that fall below this threshold in functionality. Moreover meeting this bar might remain difficult as it has been shown [3, 9, 4] that achieving functional encryption for this level of functionality is as difficult as achieving indistinguishability obfuscation [7, 20].

Delaying Parameters and Semi-Adaptive Security One remarkable feature of almost all of the aforementioned works is that the security reductions treat the second key query phase identically to the first. Indeed papers will often simply describe the proof of Phase 2 key queries as being the same as Phase 1. Lewko and Waters [28] first departed from this paradigm in a proof where they gave an ABE scheme with a security reduction handled Phase 1 and Phase 2 keys differently. Central to their proof was what they called a "delayed parameters" technique that delayed the release of part of the public parameters in a way that gave a bridge for building adaptive security proofs utilizing selective type techniques. These ideas were extended and codified into a framework by Attrapadung [6].

Chen and Wee [18] introduced the definition of semi-adaptive security as a notion of security where an attacker discloses the challenge messages after it sees the public parameters, but before it makes any key queries. It is easy to see that this notion falls somewhere between selective and adaptive in terms of strength.

Most recently, Brakerski and Vaikuntanathan [16] gave an interesting circuit ABE scheme that was provably secure in the semi-adaptive model from the Learning with Error assumption [31]. Their cryptosystem and proof of security build upon the (arithmetic) circuit ABE scheme of Boneh et al. [13] and requires a somewhat elaborate two level application of these techniques integrated with a pseudorandom function (we note that some of the complexity is due to their parallel goal of bundling functionalities; we will return to

[2] There also exists simulation-based notions of security [14, 30], but these will not be a focus of this work.

[3] We note that FE for poly-sized circuits is achievable by bootstrapping FE for **NC1** [23].

this). Like the earlier work of [28], they also apply a "delayed parameter" concept, although its flavor and execution are significantly different.

1.1 Going from Selective to Semi-Adaptive Security Generically

We now arrive at the first goal of this work.

Can we generically transform any selectively secure attribute-based encryption or functional encryption scheme into one that is semi-adaptively secure?

It turns out that this transformation is possible and moreover that the method to do so is quite simple. Here is our idea in a nutshell. Instead of encrypting the data outright, the encryptor will consider a circuit that fixes the message and randomness for encryption and takes the functional encryption scheme's public parameters as input. It then garbles this circuit and encrypts each pair of input wire values under pairs of standard PKE public keys provided by the authority. The garbled circuit plus pairs of encrypted wires comprise the ciphertext. In generating a secret key, the authority will output both the underlying functional encryption secret key as well as give one of the PKE secret keys for each pair corresponding to the underlying selectively secure FE public parameters. The decryption algorithm will first evaluate the garbled circuit to obtain the underlying ciphertext and then decrypt using the FE secret key. In this manner, the core FE parameters are literally not committed to until a key is given out.

We now elaborate our description. Let $\mathsf{FE}_{\mathrm{sel}} = (\mathsf{Setup}_{\mathrm{sel}}, \mathsf{Enc}_{\mathrm{sel}}, \mathsf{KeyGen}_{\mathrm{sel}}, \mathsf{Dec}_{\mathrm{sel}})$ be the underlying selectively secure FE scheme. Our semi-adaptively secure FE setup algorithm generates a master public/secret key pair $(\mathsf{mpk}_{\mathsf{sel}}, \mathsf{msk}_{\mathsf{sel}})$ using $\mathsf{Setup}_{\mathrm{sel}}$, and chooses 2ℓ public/secret key pairs $\{\mathsf{pk}_{i,b}, \mathsf{sk}_{i,b}\}$ for a semantically secure PKE scheme, where $\ell = |\mathsf{mpk}_{\mathsf{sel}}|$. The public key of $\mathsf{FE}_{\mathrm{sel}}$ consists of these 2ℓ PKE public keys $\{\mathsf{pk}_{i,b}\}$, but *not* the public key $\mathsf{mpk}_{\mathsf{sel}}$. To encrypt any message m, the encryptor constructs a circuit that takes as input an ℓ bit string str and outputs $\mathsf{Enc}_{\mathrm{sel}}(\mathsf{str}, m; r)$ – an encryption of m using str as the public key and r as randomness. The encryptor garbles this circuit and encrypts each of the 2ℓ garbled circuit input wire keys $w_{i,b}$ under the corresponding public key $\mathsf{pk}_{i,b}$. The ciphertext consists of the garbled circuit and the 2ℓ encrypted wire keys. The secret key for any function f consists of three parts — the master public key $\mathsf{mpk}_{\mathsf{sel}}$, ℓ PKE secret keys to decrypt half of the encrypted wire keys $w_{i,b}$ corresponding to $\mathsf{mpk}_{\mathsf{sel}}$, and $\mathsf{FE}_{\mathrm{sel}}$ secret key $\mathsf{sk}_{f,\mathsf{sel}}$ to decrypt the actual $\mathsf{FE}_{\mathrm{sel}}$ ciphertext. The key $\mathsf{sk}_{f,\mathsf{sel}}$ is simply generated using the $\mathsf{KeyGen}_{\mathrm{sel}}$ algorithm, and the ℓ PKE secret keys released correspond to the bits of $\mathsf{mpk}_{\mathsf{sel}}$. For decrypting any ciphertext, the decryptor first decrypts the encrypted input wire keys. Then, these wire keys are used to evaluate the garbled circuit. This evaluation results in an $\mathsf{FE}_{\mathrm{sel}}$ ciphertext under $\mathsf{mpk}_{\mathsf{sel}}$, which can be decrypted using $\mathsf{sk}_{f,\mathsf{sel}}$.

The crucial observation here is that the underling $\mathsf{FE}_{\mathrm{sel}}$ public key $\mathsf{mpk}_{\mathsf{sel}}$ is information theoretically hidden until any secret key is given out as the encryptor computes the ciphertext oblivious to the knowledge of $\mathsf{mpk}_{\mathsf{sel}}$. Therefore, the semi-adaptive security proof follows from a simple sequence of hybrids. In the first hybrid, we switch the ℓ encryptions of input wire keys (given out in the challenge ciphertext) which are never decrypted to encryptions of zeros. Next, in the following hybrid, we simulate the garbled circuit (given out in the challenge ciphertext) instead of constructing the actual encryption circuit and garbling it. After these two indistinguishable hybrid jumps, we could directly reduce the semi-adaptive security to selective security as the $\mathsf{FE}_{\mathrm{sel}}$ public key is hidden. Our construction and security proof is described in detail in Section 4.

The overhead associated with our transformation to semi-adaptive security is readily apparent. Instead of evaluating the underlying encryption algorithm, the transformed encryption algorithm will need to garble the encryption circuit. The ciphertext will grow proportionally to the size of this garbled circuit. Similarly, the decryption algorithm will first have to evaluate the garbled circuit before executing the core decryption. In the description above one will replace each bit of the original public parameters with a pair of PKE public keys. However, if one optimizes by using IBE instead of PKE for this step, the public parameters could actually become shorter than the original ones. In many cases our transformation will incur greater overhead than non-generic techniques designed with knowledge of the underlying scheme such as [18].

Interpreting our Result It is useful to step back and see what light our result can shed on the relationship between selective, semi-adaptive and adaptive security. Ideally, we would like to claim that semi-adaptive security gives us a half-way point between selective and adaptive where the next idea could take us all the way between the two endpoints. While this might turn out to be the case, the way in which we delay parameters seems primarily to exploit the closeness of selective and semi-adaptive security as opposed to crossing a great divide. To us this suggests that the moral gap between selective and semi-adaptive security is much smaller than that between semi-adaptive and full security (at least for functionalities that fall below the threshold needed by [2]). We view illuminating this relationship as one of the contributions of this paper.

1.2 Bundling Functionalities

We now turn to the second goal of our work. Before doing so, we describe a more general definition of functional encryption, which will later help us to explain our idea of bundling functionalities. Any functional encryption scheme is associated with a message space $\mathcal{M}$ and function space $\mathcal{F}$. In many scenarios, the function space $\mathcal{F}$ and message space $\mathcal{M}$ themselves consists of a sequence of function spaces $\{\mathcal{F}_n\}_n$ and message spaces $\{\mathcal{M}_n\}_n$ respectively, parameterized by the 'functionality index' n. In our definition of functional encryption, we assume that the setup algorithm takes two inputs - the security parameter λ and the functionality index n. This notation decouples the security of the scheme from the choice of functionality it provides. We note that such terminology has appeared in several prior works. For example, Goyal et al. [25] have a setup algorithm that takes as input the number of attributes along with the security parameter. Similarly, in the works of Boyen and Waters [15] and Agrawal et al. [1], the setup algorithms also take the length of vectors as an input. And other works [24, 13] specify the maximum depth of a circuit in an ABE scheme during setup.

Using the above convention, $\mathsf{Setup}(1^\lambda, 1^n)$ creates a master public/secret key for message space $\mathcal{M}_n$ and function space $\mathcal{F}_n$. For example, in an inner product encryption scheme, the setup algorithm fixes the length of vectors to be encrypted once the master public key is fixed. However, one goal could be to allow more flexibility after the public key is published. In particular, would it be possible to have all message and function spaces available even after setup? Continuing our example, we might want an inner product encryption scheme where the encryptor/key generator are allowed to encrypt/ generate keys for arbitrary length vectors after the public parameters have been fixed.

Looking more generally, a natural question to ask is — "Can we generically transform any (standard) functional encryption scheme into one where a *single* set of public parameters can support the union of underlying message/function spaces?" We answer this in the affirmative, and show a generic transformation using identity based encryption, pseudorandom functions and garbled circuits, all of which can be realized from standard assumptions. More formally, we show how to transform an FE scheme with message space $\{\mathcal{M}_n\}_n$ and function space $\{\mathcal{F}_n\}_n$ to an FE scheme for message space $\mathcal{M} = \cup_n \mathcal{M}_n$ and function space $\mathcal{F} = \cup_n \mathcal{F}_n$. The key for a function $f \in \mathcal{F}_n$ can be used with a ciphertext for message $m \in \mathcal{M}_n$ to compute $f(m)$. If f and m are not compatible (i.e. $f \in \mathcal{F}_n$ and $m \in \mathcal{M}_{n'}$), then the decryption fails.

As a simple instantiation, using our transformation, one can construct an inner product encryption scheme where the encryption algorithm and the key generation algorithm can both take arbitrary length vectors as input. However, given a secret key for vector v and an encryption of vector w, the decryption algorithm tests orthogonality only if v and w have same length; else the decryption algorithm fails. Similarly, our transformation can also capture the recent result of Brakerski and Vaikuntanathan [16]. They give a circuit ABE scheme where under a single set of parameters an encryptor can encrypt messages for an attribute of unbounded length. Later if a private key is given out and is tied to the same attribute length it can decrypt if the circuit matches. In our transformation we would start with a selective scheme for circuit ABE such as [24] where 1^n denotes the number of attributes and then apply our transformation. We observe that we could even choose to obtain more flexibility where we might allow both the attribute length and circuit depth to depend on 1^n.

One should be careful to point out the limits of such bundling. The main restriction is that in order for decryption to do anything useful the functionality index used to encrypt must match that of the private key; otherwise they simply are not compatible. So such a technique cannot be used to emulate a functionality

such as ABE for DFAs [37] or Functional Encryption for Turning Machines [5] where the base private key is meant to operate on ciphertext attributes of unrestricted size.

Our Transformation for Bundling Functionalities Our method for achieving such a transformation follows in a similar line to the selective to semi-adaptive transformation given above. In addition, it also amplifies to semi-adaptive security along the way for free. Recall, in the base scheme, $\mathsf{Setup}_{\mathsf{sel}}$ takes functionality index n as input and outputs master public/secret keys. Let $\ell(n)$ denote the bit-length of public keys output by $\mathsf{Setup}_{\mathsf{sel}}$. In our transformed scheme, the setup algorithm chooses IBE public/secret keys $(\mathsf{mpk}_{\mathsf{IBE}}, \mathsf{msk}_{\mathsf{IBE}})$ and sets $\mathsf{mpk}_{\mathsf{IBE}}$ as the public key. To encrypt a message $m \in \mathcal{M}_n$, the encryptor first constructs a circuit which takes a $\ell = \ell(n)$ bit input string str, and then outputs $\mathsf{Enc}_{\mathsf{sel}}(\mathsf{str}, m; r)$. The encryptor then garbles this circuit, and each wire key $w_{i,b}$ is encrypted for identity (n, i, b). The final ciphertext consists of the garbled circuit, together with encryptions of wire keys. Note that ℓ is not fixed during setup, and hence we need to use IBE instead of PKE.

The secret key for a function $f \in \mathcal{F}_t$ is computed as follows. First, the key generation algorithm chooses pseudorandom $\mathsf{FE}_{\mathrm{sel}}$ keys $(\mathsf{mpk}_t, \mathsf{msk}_t)$ using $\mathsf{Setup}_{\mathsf{sel}}(1^\lambda, 1^t)$. Next, it computes IBE secret keys for identities $(t, i, \mathsf{mpk}_t[i])$. Finally, it computes an $\mathsf{FE}_{\mathrm{sel}}$ secret key for the function f. The decryption procedure is similar to the one described in Section 1.1. Let $\mathsf{ct} = (C, \{ct_{i,b}\})$ and $\mathsf{sk}_f = (\{\mathsf{sk}_i\}, \mathsf{sk}_{f,\mathsf{sel}})$. First, note that it is important that the message underlying the ciphertext, and the function underlying the secret key are compatible. If so, the decryptor first decrypts $\mathsf{ct}_{i,b}$ to compute the garbled circuit wire keys. Next, it evaluates the garbled circuit to get an $\mathsf{FE}_{\mathrm{sel}}$ ciphertext, which it then decrypts using $\mathsf{sk}_{f,\mathsf{sel}}$. The proof of security is along the lines of selective to semi-adaptive transformation proof.

The overhead involved in this transformation is similar to the overhead in going from selective to semi-adaptive security, except that the size of the garbled circuit, and the number of wire keys grows with the functionality index. Overhead comparisons between our approach and the non-generic approach of [16] are less clear, since their approach requires increasing the maximum depth of the circuit to accommodate a PRF evaluation before evaluating the main circuit.

2 Preliminaries

2.1 Garbled Circuits

Our definition of garbled circuits [40] is based upon the work of Bellare et al. [8]. Let $\{\mathcal{C}_n\}_n$ be a family of circuits where each circuit in $\mathcal{C}_n$ takes n bit inputs. A garbling scheme GC for circuit family $\{\mathcal{C}_n\}_n$ consists of polynomial-time algorithms Garble and Eval with the following syntax.

- $\mathsf{Garble}(C \in \mathcal{C}_n, 1^\lambda)$: The garbling algorithm takes as input the security parameter λ and a circuit $C \in \mathcal{C}_n$. It outputs a garbled circuit G, together with $2n$ wire keys $\{w_{i,b}\}_{i \le n, b \in \{0,1\}}$.
- $\mathsf{Eval}(G, \{w_i\}_{i \le n})$: The evaluation algorithm takes as input a garbled circuit G and n wire keys $\{w_i\}_{i \le n}$ and outputs $y \in \{0, 1\}$.

Correctness: A garbling scheme GC for circuit family $\{\mathcal{C}_n\}_n$ is said to be correct if for all λ, n, $x \in \{0,1\}^n$ and $C \in \mathcal{C}_n$, $\mathsf{Eval}(G, \{w_{i,x_i}\}_{i \le n}) = C(x)$, where $(G, \{w_{i,b}\}_{i \le n, b \in \{0,1\}}) \leftarrow \mathsf{Garble}(C, 1^\lambda)$.

Security: Informally, a garbling scheme is said to be secure if for every circuit C and input x, the garbled circuit G together with input wires $\{w_{i,x_i}\}_{i \le n}$ corresponding to some input x reveals only the output of the circuit $C(x)$, and nothing else about the circuit C or input x.

Definition 2.1. A garbling scheme $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Eval})$ for a class of circuits $\mathcal{C} = \{\mathcal{C}_n\}_n$ is said to be a secure garbling scheme if there exists a polynomial-time simulator Sim such that for all λ, n, $C \in \mathcal{C}_n$ and $x \in \{0,1\}^n$, the following holds:

$$\left\{\mathsf{Sim}\left(1^\lambda, 1^n, 1^{|C|}, C(x)\right)\right\} \approx_c \left\{(G, \{w_{i,x_i}\}_{i\le n}) \;:\; \left(G, \{w_{i,b}\}_{i\le n, b\in\{0,1\}}\right) \leftarrow \mathsf{Garble}(C, 1^\lambda)\right\}.$$

2.2 Public Key Encryption

A Public Key Encryption (PKE) scheme $\mathsf{PKE} = (\mathsf{Setup}_{\mathrm{PKE}}, \mathsf{Enc}_{\mathrm{PKE}}, \mathsf{Dec}_{\mathrm{PKE}})$ with message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_\lambda$ consists of the following polynomial-time algorithms:

- $\mathsf{Setup}_{\mathrm{PKE}}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$: The setup algorithm is a randomized algorithm that takes security parameter λ as input and outputs a public-secret key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Enc}_{\mathrm{PKE}}(\mathsf{pk}, m \in \mathcal{M}_\lambda) \to \mathsf{ct}$: The encryption algorithm is a randomized algorithm that takes as inputs the public key pk, and a message m and outputs a ciphertext ct.
- $\mathsf{Dec}_{\mathrm{PKE}}(\mathsf{sk}, \mathsf{ct}) \to \mathcal{M}_\lambda$: The decryption algorithm is a deterministic algorithm that takes as inputs the secret key sk, and a ciphertext ct and outputs a message m.

Correctness : For correctness, we require that for all $\lambda \in \mathbb{N}$, $m \in \mathcal{M}_\lambda$, and $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}_{\mathrm{PKE}}(1^\lambda)$,

$$\Pr[\mathsf{Dec}_{\mathrm{PKE}}(\mathsf{sk}, \mathsf{Enc}_{\mathrm{PKE}}(\mathsf{pk}, m)) = m] = 1.$$

Security : For security, we require PKE to be semantically secure, i.e. the adversary must not be able to distinguish between encryptions of distinct messages of its own choosing even after receiving the public key. The notion of semantical security for PKE schemes is defined below.

Definition 2.2. A PKE scheme $\mathsf{PKE} = (\mathsf{Setup}_{\mathrm{PKE}}, \mathsf{Enc}_{\mathrm{PKE}}, \mathsf{Dec}_{\mathrm{PKE}})$ is said to be semantically secure if there exists $\lambda_0 \in \mathbb{N}$ such that for every PPT attacker $\mathcal{A}$ there exists a negligible function $\mathrm{negl}(\cdot)$ such that for all $\lambda \ge \lambda_0$, $\mathsf{Adv}^{\mathsf{PKE}}_{\mathcal{A}}(\lambda) = |\Pr[\mathsf{Exp\text{-}PKE}(\mathsf{PKE}, \lambda, \mathcal{A}) = 1] - 1/2| \le \mathrm{negl}(\lambda)$, where $\mathsf{Exp\text{-}PKE}$ is defined in Figure 1.

2.3 Identity-Based Encryption

An Identity-Based Encryption (IBE) scheme $\mathsf{IBE} = (\mathsf{Setup}_{\mathrm{IBE}}, \mathsf{KeyGen}_{\mathrm{IBE}}, \mathsf{Enc}_{\mathrm{IBE}}, \mathsf{Dec}_{\mathrm{IBE}})$ with message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_\lambda$ and identity space $\mathcal{I} = \{\mathcal{I}_\lambda\}_\lambda$ consists of the following polynomial-time algorithms:

- $\mathsf{Setup}_{\mathrm{IBE}}(1^\lambda) \to (\mathsf{pp}, \mathsf{msk})$: The setup algorithm is a randomized algorithm that takes security parameter λ as input and outputs $(\mathsf{pp}, \mathsf{msk})$, where pp are public parameters and msk is the master secret key.
- $\mathsf{KeyGen}_{\mathrm{IBE}}(\mathsf{msk}, \mathsf{ID} \in \mathcal{I}_\lambda) \to \mathsf{sk}_{\mathsf{ID}}$: The key generation algorithm is a randomized algorithm that takes as inputs the master secret key msk, and an identity ID and outputs a secret key $\mathsf{sk}_{\mathsf{ID}}$.
- $\mathsf{Enc}_{\mathrm{IBE}}(\mathsf{pp}, m \in \mathcal{M}_\lambda, \mathsf{ID} \in \mathcal{I}_\lambda) \to \mathsf{ct}$: The encryption algorithm is a randomized algorithm that takes as inputs the public parameters pp, a message m, and an identity ID and outputs a ciphertext ct.
- $\mathsf{Dec}_{\mathrm{IBE}}(\mathsf{sk}_{\mathsf{ID}}, \mathsf{ct}) \to \mathcal{M}_\lambda \cup \{\perp\}$: The decryption algorithm is a deterministic algorithm that takes as inputs the secret key $\mathsf{sk}_{\mathsf{ID}}$, and a ciphertext ct and outputs a message m or $\perp$.

Correctness : For correctness, we require that for all $\lambda \in \mathbb{N}$, $m \in \mathcal{M}_\lambda$, $\mathsf{ID} \in \mathcal{I}_\lambda$, and $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}_{\mathrm{IBE}}(1^\lambda)$,

$$\Pr[\mathsf{Dec}_{\mathrm{IBE}}(\mathsf{KeyGen}_{\mathrm{IBE}}(\mathsf{msk}, \mathsf{ID}), \mathsf{Enc}_{\mathrm{IBE}}(\mathsf{pp}, m, \mathsf{ID})) = m] = 1.$$

$\underline{\mathsf{Exp\text{-}PKE}(\mathsf{PKE}, \lambda, \mathcal{A})}$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}_{\mathrm{PKE}}(1^\lambda)$
$(m_0^*, m_1^*) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pk})$
$b \leftarrow \{0,1\}, \mathsf{ct}^* \leftarrow \mathsf{Enc}_{\mathrm{PKE}}(\mathsf{pk}, m_b^*)$
$b' \leftarrow \mathcal{A}(\mathsf{ct}^*)$
Output $(b' \stackrel{?}{=} b)$

$\underline{\mathsf{Exp\text{-}IBE}(\mathsf{IBE}, \lambda, \mathcal{A})}$
$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}_{\mathrm{IBE}}(1^\lambda)$
$(m_0^*, m_1^*, \mathsf{ID}^*) \leftarrow \mathcal{A}^{\mathsf{KeyGen}_{\mathrm{IBE}}(\mathsf{msk}, \cdot)}(1^\lambda, \mathsf{mpk})$
$b \leftarrow \{0,1\}, \mathsf{ct}^* \leftarrow \mathsf{Enc}_{\mathrm{IBE}}(\mathsf{mpk}, m_b^*, \mathsf{ID}^*)$
$b' \leftarrow \mathcal{A}^{\mathsf{KeyGen}_{\mathrm{IBE}}(\mathsf{msk}, \cdot)}(\mathsf{ct}^*)$
Output $(b' \stackrel{?}{=} b)$

Figure 1: The PKE and IBE security games. In both the games, we assume that the adversary $\mathcal{A}$ is stateful. And in the IBE security game, we also require that ID^* is not queried to the key generation oracle.

Security : For security, intuitively, we require that if an adversary has keys for identities $\{\mathsf{ID}_i\}_i$, and ct is a ciphertext for identity $\mathsf{ID}^* \neq \mathsf{ID}_i$ for all i, then the adversary must not be able to recover the underlying message. This is formally defined via the following security game between a challenger and an adversary.

Definition 2.3. An IBE scheme $\mathsf{IBE} = (\mathsf{Setup}_{\mathrm{IBE}}, \mathsf{KeyGen}_{\mathrm{IBE}}, \mathsf{Enc}_{\mathrm{IBE}}, \mathsf{Dec}_{\mathrm{IBE}})$ is said to be fully secure if there exists $\lambda_0 \in \mathbb{N}$ such that for every PPT attacker $\mathcal{A}$ there exists a negligible function $\mathrm{negl}(\cdot)$ such that for all $\lambda \geq \lambda_0$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{IBE}}(\lambda) = |\Pr[\mathsf{Exp\text{-}IBE}(\mathsf{IBE}, \lambda, \mathcal{A}) = 1] - 1/2| \leq \mathrm{negl}(\lambda)$, where $\mathsf{Exp\text{-}IBE}$ is defined in Figure 1.

3 Functional Encryption

The notion of functional encryption was formally defined in the works of Boneh, Sahai and Waters[14] and O'Neill[30]. A functional encryption scheme consists of a setup algorithm, an encryption algorithm, a key generation algorithm and a decryption algorithm. The setup algorithm takes the security parameter as input and outputs a public key and a master secret key. The encryption algorithm uses the public key to encrypt a message, while the key generation algorithm uses the master secret key to compute a secret key corresponding to a function. The decryption algorithm takes as input a ciphertext and a secret key, and outputs the function evaluation on the message.

The functionality index : Every functional encryption scheme is associated with a message space which defines the set of messages that can be encrypted, and a function space which defines the set of functions for which a secret key can be generated. In most schemes, the message space $\mathcal{M}$ and the function space $\mathcal{F}$ consists of a sequence of message spaces $\{\mathcal{M}_n\}_{n \in \mathbb{N}}$ and function spaces $\{\mathcal{F}_n\}_n$, both parameterized by the functionality index (the special case where $\mathcal{M}_n = \mathcal{M}$ and $\mathcal{F}_n = \mathcal{F}$ for all $n \in \mathbb{N}$ is discussed in Section 3.1).

The choice of functionality index: A minor definitional issue that arises is with respect to the choice of functionality index. Some works use the security parameter itself to define a message space $\mathcal{M}_\lambda$ and function space $\mathcal{F}_\lambda$. For example, in the inner product FE scheme of Katz et al. [26], the message space and function space are set to be $\mathbb{Z}_q^\lambda$ during setup, where λ is the security parameter and q is an appropriately chosen modulus.

A more flexible approach is to decouple the security parameter and the *functionality index*, and allow the setup algorithm to take two inputs - a security parameter λ and a functionality index n. This additional parameter then defines the message space for the encryption algorithm and the function space for the key generation algorithm. Some existing works implicitly assume that the setup algorithm also receives such a parameter as input. For example, in the work of Goyal et al. [25], the universe $\mathcal{U} = \{1, 2, \ldots, n\}$ is defined as the universe of attributes for the ABE scheme. Other works, such as the inner product FE scheme of Agrawal et al. [1] explicitly mention this as an input to the setup algorithm. We will also use this approach in our formal definition of a functional encryption scheme.

Formal definition : Let $\mathcal{M} = \{\mathcal{M}_n\}_{n\in\mathbb{N}}$, $\mathcal{R} = \{\mathcal{R}_n\}_{n\in\mathbb{N}}$ be families of sets, and $\mathcal{F} = \{\mathcal{F}_n\}$ a family of functions, where for all $n \in \mathbb{N}$ and $f \in \mathcal{F}_n$, $f : \mathcal{M}_n \to \mathcal{R}_n$. We will also assume that for all $n \in \mathbb{N}$, the set $\mathcal{F}_n$ contains an *empty function* $\epsilon_n : \mathcal{M}_n \to \mathcal{R}_n$. As in [14], the empty function is used to capture information that intentionally leaks from the ciphertext. For instance, in a PKE scheme, the length of the message could be revealed from the ciphertext. Similarly, in an attribute based encryption scheme, the ciphertext could reveal the attribute for which the message was encrypted.

A functional encryption scheme FE for function space $\{\mathcal{F}_n\}_{n\in\mathbb{N}}$ and message space $\{\mathcal{M}_n\}_{n\in\mathbb{N}}$ consists of four polynomial-time algorithms (Setup, Enc, KeyGen, Dec) with the following syntax.

- $\mathsf{Setup}(1^\lambda, 1^n) \to (\mathsf{mpk}, \mathsf{msk})$: The setup algorithm is a randomized algorithm that takes as input the security parameter λ and the functionality index n, and outputs the master public/secret key pair $(\mathsf{mpk}, \mathsf{msk})$.
- $\mathsf{Enc}(\mathsf{mpk}, m \in \mathcal{M}_n) \to \mathsf{ct}$: The encryption algorithm is a randomized algorithm that takes as input the public key mpk and a message $m \in \mathcal{M}_n$ and outputs a ciphertext ct.
- $\mathsf{KeyGen}(\mathsf{msk}, f \in \mathcal{F}_n) \to \mathsf{sk}_f$: The key generation algorithm is a randomized algorithm that takes as input the master secret key msk and a function $f \in \mathcal{F}_n$ and outputs a secret key sk_f.
- $\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}) \to \{0, 1, \perp\}$: The decryption algorithm is deterministic. It takes as input a ciphertext ct and a secret key sk_f and outputs $y \in \{0, 1, \perp\}$.

More general definitions of functional encryption: It is possible to consider more general definitions for functional encryption. For example, one could consider a definition where the setup algorithm takes as input a security parameter λ, functionality index n and a depth-index d that bounds the circuit depth of $\mathcal{F}_n$. For simplicity of notation we avoid such extensions, although we believe that our results can be generalized for all such extensions.

Correctness: A functional encryption scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ is said to be correct if for all security parameter λ and functionality index n, functions $f \in \mathcal{F}_n$, messages $m \in \mathcal{M}_n$ such that (f, m) are compatible, and $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$,

$$\Pr\left[\mathsf{Dec}(\mathsf{KeyGen}(\mathsf{msk}, f), \mathsf{Enc}(\mathsf{mpk}, m)) = f(m)\right] = 1.$$

Security: Informally, a functional encryption scheme is said to be secure if an adversary having secret keys for functions $\{f_i\}_{i\le k}$ and a ciphertext ct for message m learns only $\{f_i(m)\}_{i\le k}$, and nothing else about the underlying message m. This can be formally captured via the following 'indistinguishability based' security definition.

Definition 3.1. A functional encryption scheme FE is adaptively secure if there exists $\lambda_0 \in \mathbb{N}$ such that for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathrm{negl}(\cdot)$ such that for all $\lambda > \lambda_0$, $n \in \mathbb{N}$, $|\Pr[\mathsf{Exp\text{-}adaptive}(\mathsf{FE}, \lambda, n, \mathcal{A}) = 1] - 1/2| \le \mathrm{negl}(\lambda)$, where $\mathsf{Exp\text{-}adaptive}$ is defined in Figure 2.

A weaker notion of security is that of selective security, where the adversary must declare the challenge inputs before receiving the public parameters.

Definition 3.2. A functional encryption scheme FE is selectively secure if there exists $\lambda_0 \in \mathbb{N}$ such that for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathrm{negl}(\cdot)$ such that for all $\lambda > \lambda_0$, $n \in \mathbb{N}$, $|\Pr[\mathsf{Exp\text{-}selective}(\mathsf{FE}, \lambda, n, \mathcal{A}) = 1] - 1/2| \le \mathrm{negl}(\lambda)$, where $\mathsf{Exp\text{-}selective}$ is defined in Figure 2.

Finally, we have an intermediate notion of security called semi-adaptive security, where the adversary must declare the challenge inputs before receiving any key queries.

Definition 3.3. A functional encryption scheme FE is semi-adaptively secure if there exists $\lambda_0 \in \mathbb{N}$ such that for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathrm{negl}(\cdot)$ such that for all $\lambda > \lambda_0$, $n \in \mathbb{N}$, $|\Pr[\mathsf{Exp\text{-}semi\text{-}adp}(\mathsf{FE}, \lambda, n, \mathcal{A}) = 1] - 1/2| \le \mathrm{negl}(\lambda)$, where $\mathsf{Exp\text{-}semi\text{-}adp}$ is defined in Figure 2.

$\underline{\mathsf{Exp\text{-}adaptive}(\mathsf{FE}, \lambda, n, \mathcal{A})}$
$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$
$(m_0^*, m_1^*) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(1^\lambda, \mathsf{mpk})$
$b \leftarrow \{0,1\}, \mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, m_b^*)$
$b' \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{ct}^*)$
Output $(b' \stackrel{?}{=} b)$

$\underline{\mathsf{Exp\text{-}selective}(\mathsf{FE}, \lambda, n, \mathcal{A})}$
$(m_0^*, m_1^*) \leftarrow \mathcal{A}(1^\lambda)$
$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$
$b \leftarrow \{0,1\}, \mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, m_b^*)$
$b' \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{mpk}, \mathsf{ct}^*)$
Output $(b' \stackrel{?}{=} b)$

$\underline{\mathsf{Exp\text{-}semi\text{-}adp}(\mathsf{FE}, \lambda, n, \mathcal{A})}$
$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$
$(m_0^*, m_1^*) \leftarrow \mathcal{A}(1^\lambda, \mathsf{mpk})$
$b \leftarrow \{0,1\}, \mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, m_b^*)$
$b' \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{ct}^*)$
Output $(b' \stackrel{?}{=} b)$

Figure 2: Experiments referred in Definitions 3.1, 3.2 and 3.3. We assume that the adversary $\mathcal{A}$ is stateful, $\epsilon_n(m_0^*) = \epsilon_n(m_1^*)$, and for all key queries f queried by $\mathcal{A}$ to KeyGen oracle, $f \in \mathcal{F}_n$ and $f(m_0^*) = f(m_1^*)$.

3.1 Functional Encryption with Uniform Function and Message Space

In the previous section, we saw a definition for functional encryption schemes where the setup algorithm takes the functionality index n as input, and outputs a master public/secret key pair specific to the functionality index n. As a result, the encryption algorithm, when using this public key, can only encrypt messages in the message space $\mathcal{M}_n$. Similarly, the key generation algorithm can only generate keys in the function space $\mathcal{F}_n$.

However, if there is exactly one message space, and exactly one function space (that is $\mathcal{M}_n = \mathcal{M}$ and $\mathcal{F}_n = \mathcal{F}$ for all n), then we can assume the setup algorithm takes only the security parameter as input. The remaining syntax, correctness and security definitions are same as before.

4 Selective to Semi-Adaptive Security Generically

In this section, we show how to construct semi-adaptively secure functional encryption schemes from selectively secure functional encryption schemes, semantically secure public key encryption schemes, and secure garbled circuits. At a high-level, the idea is to delay the release of the base FE scheme's public parameters until the adversary makes first key query, and since in a semi-adaptive security game the adversary must submit its challenge before requesting any secret keys, therefore we could hope to invoke the selective security of the underlying FE scheme after receiving the challenge. However, the simulator needs to provide enough information to the adversary so that it could still perform encryptions before sending the challenge. To get around this problem, the encryption algorithm is modified to output a garbled circuit which takes as input the $\mathsf{FE}_{\mathrm{sel}}$ public parameters and outputs the appropriate ciphertext. Essentially, the encryption algorithm encrypts without knowing the 'real' public parameters. The encryption algorithm would still need to hide the input wire keys such that a secret key reveals only half of them. Below we describe our approach in detail.

4.1 Construction

Let $\mathsf{FE}_{\mathrm{sel}} = (\mathsf{Setup}_{\mathrm{sel}}, \mathsf{KeyGen}_{\mathrm{sel}}, \mathsf{Enc}_{\mathrm{sel}}, \mathsf{Dec}_{\mathrm{sel}})$ be a functional encryption scheme with function space $\{\mathcal{F}_n\}_n$ and message space $\{\mathcal{M}_n\}_n$. We use the polynomial $\ell(\lambda, n)$ to denote the size of the public key output by the $\mathsf{FE}_{\mathrm{sel}}$ setup algorithm, where λ is the security parameter and n is the functionality index. We will simply write it as ℓ whenever clear from context.

Tools required for our transformation : Let $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Eval})$ be a garbling scheme for circuit family $\mathcal{C} = \{\mathcal{C}_m\}_m$, and $\mathsf{PKE} = (\mathsf{Setup}_{\mathrm{PKE}}, \mathsf{Enc}_{\mathrm{PKE}}, \mathsf{Dec}_{\mathrm{PKE}})$ be a public key encryption scheme.

Our transformation : We now describe our construction for semi-adaptively secure functional encryption scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{KeyGen})$ with message space $\{\mathcal{M}_n\}_n$ and function space $\{\mathcal{F}_n\}_n$.

- $\mathsf{Setup}(1^\lambda, 1^n) \to (\mathsf{mpk}, \mathsf{msk})$: The setup algorithm first runs the PKE setup to compute 2ℓ public/secret key pairs $\left(\mathsf{pk}_{i,b}, \mathsf{sk}_{i,b}\right)_{i \le \ell, b \in \{0,1\}} \leftarrow \mathsf{Setup}_{\mathrm{PKE}}(1^\lambda)$, independently and uniformly. It also runs $\mathsf{FE}_{\mathrm{sel}}$

setup algorithm and generates master public/secret key pair $(\mathsf{mpk}_{\mathsf{sel}}, \mathsf{msk}_{\mathsf{sel}}) \leftarrow \mathsf{Setup}_{\mathrm{sel}}(1^\lambda, 1^n)$. It sets $\mathsf{mpk} = \left\{\mathsf{pk}_{i,b}\right\}_{i\le\ell, b\in\{0,1\}}$ and $\mathsf{msk} = \left(\mathsf{mpk}_{\mathsf{sel}}, \mathsf{msk}_{\mathsf{sel}}, \{\mathsf{sk}_{i,b}\}_{i\le\ell,b\in\{0,1\}}\right)$.

- $\mathsf{Enc}(\mathsf{mpk}, m \in \mathcal{M}_n) \to \mathsf{ct}$: Let $\mathcal{C}\text{-}\mathsf{Enc}\text{-}\mathsf{pk}^{\ell}_{m,r}$ be the canonical circuit which has message m and randomness r hardwired, takes an ℓ bit input x and computes $\mathsf{Enc}_{\mathrm{sel}}(x, m; r)$; that is, it uses the input as a public key for the base FE scheme and encrypts message m using randomness r.

  The encryption algorithm constructs the circuit $\mathcal{C}\text{-}\mathsf{Enc}\text{-}\mathsf{pk}^{\ell}_{m,r}$ using uniform randomness r, and it computes the garbled circuit as $(C, \{w_{i,b}\}_{i\le\ell,b\in\{0,1\}}) \leftarrow \mathsf{Garble}(\mathcal{C}\text{-}\mathsf{Enc}\text{-}\mathsf{pk}^{\ell}_{m,r}, 1^\lambda)$. It then encrypts the garbled wire keys by computing $\mathsf{ct}_{i,b} \leftarrow \mathsf{Enc}_{\mathrm{PKE}}(\mathsf{pk}_{i,b}, w_{i,b})$ for $i \le \ell$ and $b \in \{0,1\}$, where $\mathsf{mpk} = \left\{\mathsf{pk}_{i,b}\right\}_{i\le\ell,b\in\{0,1\}}$. Finally, it outputs a ciphertext ct which consists of the garbled circuit C and the 2ℓ ciphertexts $\{\mathsf{ct}_{i,b}\}_{i\le\ell,b\in\{0,1\}}$.

- $\mathsf{KeyGen}(\mathsf{msk}, f \in \mathcal{F}_n) \to \mathsf{sk}_f$: Let $\mathsf{msk} = \left(\mathsf{mpk}_{\mathsf{sel}}, \mathsf{msk}_{\mathsf{sel}}, \{\mathsf{sk}_{i,b}\}_{i\le\ell,b\in\{0,1\}}\right)$. The key generation algorithm first generates selective FE secret key corresponding to the function f by computing $\mathsf{sk}_{f,\mathsf{sel}} \leftarrow \mathsf{KeyGen}_{\mathrm{sel}}(\mathsf{mpk}_{\mathsf{sel}}, f)$. It outputs $\mathsf{sk}_f = \left(\mathsf{mpk}_{\mathsf{sel}}, \mathsf{sk}_{f,\mathsf{sel}}, \left\{\mathsf{sk}_{i,\mathsf{mpk}_{\mathsf{sel}}[i]}\right\}_{i\le\ell}\right)$ as the key for function f.

- $\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}) \to \{0, 1, \perp\}$: Let $\mathsf{sk}_f = \left(\mathsf{mpk}_{\mathsf{sel}}, \mathsf{sk}_{f,\mathsf{sel}}, \{\mathsf{sk}_i\}_{i\le\ell}\right)$ and $\mathsf{ct} = \left(C, \{\mathsf{ct}_{i,b}\}_{i\le\ell,b\in\{0,1\}}\right)$. The decryption algorithm first decrypts the appropriate garbled circuit wires. Concretely, for $i \le \ell$, it computes $w_i = \mathsf{Dec}_{\mathrm{PKE}}(\mathsf{sk}_i, \mathsf{ct}_{i,\mathsf{mpk}_{\mathsf{sel}}[i]})$. It then uses these ℓ wire keys to evaluate the garbled circuit as $\widetilde{\mathsf{ct}} = \mathsf{Eval}(C, \{w_i\}_{i\le\ell})$. Finally, it uses the secret key $\mathsf{sk}_{f,\mathsf{sel}}$ to decrypt the ciphertext $\widetilde{\mathsf{ct}}$, and outputs $\mathsf{Dec}_{\mathrm{sel}}(\mathsf{sk}_{f,\mathsf{sel}}, \widetilde{\mathsf{ct}})$.

Correctness. For all $\lambda, n \in \mathbb{N}$, message $m \in \mathcal{M}_n$, base FE keys $(\mathsf{mpk}_{\mathsf{sel}}, \mathsf{msk}_{\mathsf{sel}}) \leftarrow \mathsf{Setup}_{\mathrm{sel}}(1^\lambda, 1^n)$, and 2ℓ PKE keys $\left(\mathsf{pk}_{i,b}, \mathsf{sk}_{i,b}\right) \leftarrow \mathsf{Setup}_{\mathrm{PKE}}(1^\lambda)$, the ciphertext corresponding to message m in our FE scheme is $(C, \{\mathsf{ct}_{i,b}\})$, where $(C, \{w_{i,b}\}) \leftarrow \mathsf{Garble}(\mathcal{C}\text{-}\mathsf{Enc}\text{-}\mathsf{pk}^{\ell}_{m,r}, 1^\lambda)$ and $\mathsf{ct}_{i,b} \leftarrow \mathsf{Enc}_{\mathrm{PKE}}(\mathsf{pk}_{i,b}, w_{i,b})$.

For any function $f \in \mathcal{F}_n$, the corresponding secret key in our scheme consists of $\left(\mathsf{mpk}_{\mathsf{sel}}, \mathsf{sk}_{f,\mathsf{sel}}, \left\{\mathsf{sk}_{i,\mathsf{mpk}_{\mathsf{sel}}[i]}\right\}\right)$, where $\mathsf{sk}_{f,\mathsf{sel}} \leftarrow \mathsf{KeyGen}_{\mathrm{sel}}(\mathsf{msk}_{\mathsf{sel}}, f)$. The decryption algorithm first decrypts the encryptions of garbled circuit input wires corresponding to the public key $\mathsf{mpk}_{\mathsf{sel}}$ as $w_{i,\mathsf{mpk}_{\mathsf{sel}}[i]} = \mathsf{Dec}_{\mathrm{sel}}(\mathsf{sk}_{i,\mathsf{mpk}_{\mathsf{sel}}[i]}, \mathsf{ct}_{i,\mathsf{mpk}_{\mathsf{sel}}[i]})$. This follows from correctness of PKE scheme. Next, it computes $\widetilde{\mathsf{ct}} = \mathsf{Eval}(C, \{w_{i,\mathsf{mpk}_{\mathsf{sel}}[i]}\})$ which is same as $\mathsf{Enc}_{\mathrm{sel}}(\mathsf{mpk}_{\mathsf{sel}}, m; r)$ due to correctness of garbling scheme. Finally, the decryption algorithm computes $\mathsf{Dec}_{\mathrm{sel}}(\mathsf{sk}_{f,\mathsf{sel}}, \widetilde{\mathsf{ct}})$ which is equal to $f(m)$ as the base FE scheme is also correct. Therefore, FE satisfies the functional encryption correctness condition.

Security We will now show that the scheme described above is semi-adaptively secure.

Theorem 4.1. Assuming $\mathsf{FE}_{\mathrm{sel}} = (\mathsf{Setup}_{\mathrm{sel}}, \mathsf{KeyGen}_{\mathrm{sel}}, \mathsf{Enc}_{\mathrm{sel}}, \mathsf{Dec}_{\mathrm{sel}})$ is a selectively-secure functional encryption scheme with $\{\mathcal{F}_n\}_n$ and $\{\mathcal{M}_n\}_n$ as function space and message space satisfying Definition 3.2, $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Eval})$ is a secure garbling scheme for circuit family $\mathcal{C} = \{\mathcal{C}_m\}_m$ satisfying Definition 2.1, and $\mathsf{PKE} = (\mathsf{Setup}_{\mathrm{PKE}}, \mathsf{Enc}_{\mathrm{PKE}}, \mathsf{Dec}_{\mathrm{PKE}})$ is a semantically secure public key encryption scheme satisfying Definition 2.2, then FE forms a semi-adaptively secure functional encryption scheme satisfying Definition 3.3 for same function space and message space as the selective scheme.

To formally prove our theorem, we describe the following sequence of games.

Game 1: This is the semi-adaptive security game described in Figure 2.

1. (Setup Phase) The challenger first runs the PKE setup algorithm and $\mathsf{FE}_{\mathrm{sel}}$ setup algorithm to generate public/secret key pairs as $\left(\mathsf{pk}_{i,\beta}, \mathsf{sk}_{i,\beta}\right)_{i\leq \ell, \beta\in\{0,1\}} \leftarrow \mathsf{Setup}_{\mathrm{PKE}}(1^\lambda)$ and $(\mathsf{mpk}_{\mathsf{sel}}, \mathsf{msk}_{\mathsf{sel}}) \leftarrow \mathsf{Setup}_{\mathrm{sel}}(1^\lambda, 1^n)$. It sets $\mathsf{mpk} = \left\{\mathsf{pk}_{i,\beta}\right\}_{i\leq \ell, \beta\in\{0,1\}}$ and $\mathsf{msk} = \left(\mathsf{mpk}_{\mathsf{sel}}, \mathsf{msk}_{\mathsf{sel}}, \{\mathsf{sk}_{i,\beta}\}_{i\leq \ell, \beta\in\{0,1\}}\right)$, and sends mpk to $\mathcal{A}$.
2. (Challenge Phase)
   (a) $\mathcal{A}$ sends two challenge messages (m_0^*, m_1^*) to the challenger such that $\epsilon_n(m_0^*) = \epsilon_n(m_1^*)$, where $\epsilon_n(\cdot)$ is the empty function.
   (b) Challenger chooses a random bit $b \leftarrow \{0,1\}$, and computes the garbled circuit as $(C, \{w_{i,\beta}\}_{i\leq \ell, \beta\in\{0,1\}}) \leftarrow \mathsf{Garble}(\mathcal{C}\text{-}\mathsf{Enc}\text{-}\mathsf{pk}^{\ell}_{m_b^*, r}, 1^\lambda)$.
   (c) It encrypts the wire keys $w_{i,\beta}$ as $\mathsf{ct}^*_{i,\beta} \leftarrow \mathsf{Enc}_{\mathrm{PKE}}(\mathsf{pk}_{i,\beta}, w_{i,\beta})$.
   (d) It sets challenge ciphertext as $\mathsf{ct}^* = \left(C, \left\{\mathsf{ct}^*_{i,\beta}\right\}_{i\leq \ell, \beta\in\{0,1\}}\right)$, and sends ct^* to $\mathcal{A}$.
3. (Key Query Phase)
   (a) $\mathcal{A}$ queries the challenger on polynomially many functions $f \in \mathcal{F}_n$ such that $f(m_0^*) = f(m_1^*)$.
   (b) For each queried function f, challenger generates the selective FE secret key as $\mathsf{sk}_{f,\mathsf{sel}} \leftarrow \mathsf{KeyGen}_{\mathrm{sel}}(\mathsf{msk}_{\mathsf{sel}}, f)$. It sets the secret key as $\mathsf{sk}_f = \left(\mathsf{mpk}_{\mathsf{sel}}, \mathsf{sk}_{f,\mathsf{sel}}, \left\{\mathsf{sk}_{i,\mathsf{mpk}_{\mathsf{sel}}[i]}\right\}_{i\leq \ell}\right)$, and sends sk_f to $\mathcal{A}$.
4. (Guess) Finally, $\mathcal{A}$ sends its guess b' and wins if $b = b'$.

Game 2: It is same as Game 1, except the way challenge ciphertext is created. In this game, while creating ct^*, challenger only encrypts garbled circuit wire keys corresponding to the bits of $\mathsf{mpk}_{\mathsf{sel}}$ and encrypts $\mathbf{0}$ at all other places.

1. (Setup Phase) The challenger first runs the PKE setup algorithm and $\mathsf{FE}_{\mathrm{sel}}$ setup algorithm to generate public/secret key pairs as $\left(\mathsf{pk}_{i,\beta}, \mathsf{sk}_{i,\beta}\right)_{i\leq \ell, \beta\in\{0,1\}} \leftarrow \mathsf{Setup}_{\mathrm{PKE}}(1^\lambda)$ and $(\mathsf{mpk}_{\mathsf{sel}}, \mathsf{msk}_{\mathsf{sel}}) \leftarrow \mathsf{Setup}_{\mathrm{sel}}(1^\lambda, 1^n)$. It sets $\mathsf{mpk} = \left\{\mathsf{pk}_{i,\beta}\right\}_{i\leq \ell, \beta\in\{0,1\}}$ and $\mathsf{msk} = \left(\mathsf{mpk}_{\mathsf{sel}}, \mathsf{msk}_{\mathsf{sel}}, \{\mathsf{sk}_{i,\beta}\}_{i\leq \ell, \beta\in\{0,1\}}\right)$, and sends mpk to $\mathcal{A}$.
2. (Challenge Phase)
   (a) $\mathcal{A}$ sends two challenge messages (m_0^*, m_1^*) to the challenger such that $\epsilon_n(m_0^*) = \epsilon_n(m_1^*)$, where $\epsilon_n(\cdot)$ is the empty function.
   (b) Challenger chooses a random bit $b \leftarrow \{0,1\}$, and computes the garbled circuit as $(C, \{w_{i,\beta}\}_{i\leq \ell, \beta\in\{0,1\}}) \leftarrow \mathsf{Garble}(\mathcal{C}\text{-}\mathsf{Enc}\text{-}\mathsf{pk}^{\ell}_{m_b^*, r}, 1^\lambda)$.
   (c) It then encrypts *half* of the 2ℓ wire keys as $\mathsf{ct}^*_{i,\beta} \leftarrow \mathsf{Enc}_{\mathrm{PKE}}(\mathsf{pk}_{i,\beta}, w_{i,\beta})$ if $\beta = \mathsf{mpk}_{\mathsf{sel}}[i]$, and $\mathsf{ct}^*_{i,\beta} \leftarrow \mathsf{Enc}_{\mathrm{PKE}}(\mathsf{pk}_{i,\beta}, \mathbf{0})$ otherwise.
   (d) It sets challenge ciphertext as $\mathsf{ct}^* = \left(C, \left\{\mathsf{ct}^*_{i,\beta}\right\}_{i\leq \ell, \beta\in\{0,1\}}\right)$, and sends ct^* to $\mathcal{A}$.
3. (Key Query Phase)
   (a) $\mathcal{A}$ queries the challenger on polynomially many functions $f \in \mathcal{F}_n$ such that $f(m_0^*) = f(m_1^*)$.
   (b) For each queried function f, challenger generates the selective FE secret key as $\mathsf{sk}_{f,\mathsf{sel}} \leftarrow \mathsf{KeyGen}_{\mathrm{sel}}(\mathsf{msk}_{\mathsf{sel}}, f)$. It sets the secret key as $\mathsf{sk}_f = \left(\mathsf{mpk}_{\mathsf{sel}}, \mathsf{sk}_{f,\mathsf{sel}}, \left\{\mathsf{sk}_{i,\mathsf{mpk}_{\mathsf{sel}}[i]}\right\}_{i\leq \ell}\right)$, and sends sk_f to $\mathcal{A}$.
4. (Guess) Finally, $\mathcal{A}$ sends its guess b' and wins if $b = b'$.

Game 3: It is same as Game 2, except the way challenge ciphertext is created. In this game, while creating ct^*, challenger simulates the garbled circuit instead of garbling the actual circuit.

1. (Setup Phase) The challenger first runs the PKE setup algorithm and $\mathsf{FE}_{\mathrm{sel}}$ setup algorithm to generate public/secret key pairs as $\left(\mathsf{pk}_{i,\beta}, \mathsf{sk}_{i,\beta}\right)_{i\leq \ell, \beta\in\{0,1\}} \leftarrow \mathsf{Setup}_{\mathrm{PKE}}(1^\lambda)$ and $(\mathsf{mpk}_{\mathsf{sel}}, \mathsf{msk}_{\mathsf{sel}}) \leftarrow \mathsf{Setup}_{\mathrm{sel}}(1^\lambda, 1^n)$. It sets $\mathsf{mpk} = \left\{\mathsf{pk}_{i,\beta}\right\}_{i\leq \ell, \beta\in\{0,1\}}$ and $\mathsf{msk} = \left(\mathsf{mpk}_{\mathsf{sel}}, \mathsf{msk}_{\mathsf{sel}}, \{\mathsf{sk}_{i,\beta}\}_{i\leq \ell, \beta\in\{0,1\}}\right)$, and sends mpk to $\mathcal{A}$.
2. (Challenge Phase)
   (a) $\mathcal{A}$ sends two challenge messages (m_0^*, m_1^*) to the challenger such that $\epsilon_n(m_0^*) = \epsilon_n(m_1^*)$, where $\epsilon_n(\cdot)$ is the empty function.
   (b) Challenger chooses a random bit $b \leftarrow \{0,1\}$, and computes base FE ciphertext as $\widetilde{\mathsf{ct}}^* \leftarrow \mathsf{Enc}_{\mathrm{sel}}(\mathsf{mpk}_{\mathsf{sel}}, m_b^*)$ using uniform randomness. Next, it computes the garbled circuit as $(C, \{w_{i,\mathsf{mpk}_{\mathsf{sel}}[i]}\}) \leftarrow \mathsf{Sim}\left(1^\lambda, 1^\ell, 1^k, \widetilde{\mathsf{ct}}^*\right)$, where k is the size of the canonical circuit $\mathcal{C}\text{-}\mathsf{Enc}\text{-}\mathsf{pk}^\ell_{m,r}$.
   (c) It then encrypts half of the 2ℓ wire keys as $\mathsf{ct}^*_{i,\beta} \leftarrow \mathsf{Enc}_{\mathrm{PKE}}(\mathsf{pk}_{i,\beta}, w_{i,\beta})$ if $\beta = \mathsf{mpk}_{\mathsf{sel}}[i]$, and $\mathsf{ct}^*_{i,\beta} \leftarrow \mathsf{Enc}_{\mathrm{PKE}}(\mathsf{pk}_{i,\beta}, \mathbf{0})$ otherwise.
   (d) It sets challenge ciphertext as $\mathsf{ct}^* = \left(C, \left\{\mathsf{ct}^*_{i,\beta}\right\}_{i\leq \ell, \beta\in\{0,1\}}\right)$, and sends ct^* to $\mathcal{A}$.
3. (Key Query Phase)
   (a) $\mathcal{A}$ queries the challenger on polynomially many functions $f \in \mathcal{F}_n$ such that $f(m_0^*) = f(m_1^*)$.
   (b) For each queried function f, challenger generates the selective FE secret key as $\mathsf{sk}_{f,\mathsf{sel}} \leftarrow \mathsf{KeyGen}_{\mathrm{sel}}(\mathsf{msk}_{\mathsf{sel}}, f)$. It sets the secret key as $\mathsf{sk}_f = \left(\mathsf{mpk}_{\mathsf{sel}}, \mathsf{sk}_{f,\mathsf{sel}}, \left\{\mathsf{sk}_{i,\mathsf{mpk}_{\mathsf{sel}}[i]}\right\}_{i\leq \ell}\right)$, and sends sk_f to $\mathcal{A}$.
4. (Guess) Finally, $\mathcal{A}$ sends its guess b' and wins if $b = b'$.

4.1.1 Analysis

We now establish via a sequence of lemmas that no PPT adversary can distinguish between any two adjacent games with non-negligible advantage. To conclude, we also show that any PPT adversary that wins with non-negligible probability in the last game breaks the selective security of $\mathsf{FE}_{\mathrm{sel}}$ scheme.

Let $\mathcal{A}$ be any successful PPT adversary against our construction in the semi-adaptive security game (Figure 2). In Game i, advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^i_{\mathcal{A}} = |\Pr[\mathcal{A} \text{ wins}] - 1/2|$. We then show via a sequence of claims that if $\mathcal{A}$'s advantage is non-negligible in Game i, then it has non-negligible advantage in Game $i+1$ as well. Finally, in last game, we directly use $\mathcal{A}$ to attack the selective security of underlying FE scheme. Below we describe our hybrid games in more detail.

Lemma 4.1. If PKE is a semantically secure public key encryption scheme, then for all PPT $\mathcal{A}$, $|\mathsf{Adv}^1_{\mathcal{A}} - \mathsf{Adv}^2_{\mathcal{A}}| \leq \mathrm{negl}(\lambda)$ for some negligible function $\mathrm{negl}(\cdot)$.

Proof. For proving indistinguishability of Games 1 and 2, we need to sketch ℓ intermediate hyrbrid games between these two, where ℓ is the length of master public key $\mathsf{mpk}_{\mathsf{sel}}$. Observe that in Game 1, ciphertexts $\mathsf{ct}^*_{i,\beta}$ are encryptions of garbled circuit input wire keys $w_{i,\beta}$ for both values of bit β; however, in Game 2, ciphertexts $\mathsf{ct}^*_{i,\beta}$ are encryptions of $w_{i,\beta}$ *if and only if* $\beta = \mathsf{mpk}_{\mathsf{sel}}[i]$, and they are encryptions of zeros otherwise. The high-level proof idea is to switch $\mathsf{ct}_{i,\beta}$ from encryptions of $w_{i,\beta}$ to encryptions of $\mathbf{0}$ one-at-a-time by using semantic security of PKE scheme. This could be done because the secret key $\mathsf{sk}_{i,\beta}$ is revealed only if $\beta = \mathsf{mpk}_{\mathsf{sel}}[i]$. Concretely, i^{th} intermediate hybrid between Game 1 and 2 proceeds same as Game 1 except that the first i ciphertexts $\mathsf{ct}^*_{j,\beta}$ is computed as $\mathsf{ct}^*_{j,\beta} \leftarrow \mathsf{Enc}_{\mathrm{PKE}}(\mathsf{pk}_{j,\beta}, \mathbf{0})$ if $\beta \neq \mathsf{mpk}_{\mathsf{sel}}[j]$, i.e. for $j \leq i$ and $\beta \neq \mathsf{mpk}_{\mathsf{sel}}[j]$, $\mathsf{ct}^*_{j,\beta}$ are encryptions of zero, and for $j > i$ or $\beta = \mathsf{mpk}_{\mathsf{sel}}[j]$, $\mathsf{ct}^*_{j,\beta}$ are encryptions

of wire keys $w_{i,\beta}$. For the analysis, Game 1 is regarded as 0^{th} intermediate hybrid, and Game 2 is regarded as ℓ^{th} intermediate hybrid. Below we show that $\mathcal{A}$'s advantage in distinguishing any pair of consecutive intermediate hybrid is negligibly small.

We describe a reduction algorithm $\mathcal{B}$ which breaks semantic security of the PKE scheme, if $\mathcal{A}$ distinguishes between intermediate hybrids $i-1$ and i with non-negligible advantage. First, $\mathcal{B}$ receives the challenge public key pk^* from the PKE challenger. Next, $\mathcal{B}$ runs the Step 1 as in Game 1, except instead of running PKE Setup algorithm to compute public/secret key pair $(\mathsf{pk}_{i,\beta}, \mathsf{sk}_{i,\beta})$ when $\beta \neq \mathsf{mpk}_{\mathsf{sel}}[i]$, it sets $\mathsf{pk}_{i,1-\mathsf{mpk}_{\mathsf{sel}}[i]} = \mathsf{pk}^*$. After $\mathcal{A}$ submits its challenge messages (m_0^*, m_1^*) to $\mathcal{B}$, the reduction computes garbled circuit C and ciphertexts $\mathsf{ct}_{i,\beta}^*$ as in the $(i-1)^{th}$ intermediate hybrid, except to compute $\mathsf{ct}_{i,1-\mathsf{mpk}_{\mathsf{sel}}[i]}^*$, $\mathcal{B}$ sends $w_{i,1-\mathsf{mpk}_{\mathsf{sel}}[i]}$ and $\mathbf{0}$ as its challenge messages to the PKE challenger, and sets $\mathsf{ct}_{i,1-\mathsf{mpk}_{\mathsf{sel}}[i]}^*$ as the PKE challenge ciphertext. $\mathcal{B}$ runs the remaining game as in Game 1.[4] Finally, if $\mathcal{A}$ wins ($b = b'$), then $\mathcal{B}$ guesses 0 to indicate that $\mathsf{ct}_{i,1-\mathsf{mpk}_{\mathsf{sel}}[i]}^*$ was encryption of $w_{i,1-\mathsf{mpk}_{\mathsf{sel}}[i]}$, else it guesses 1 to indicate that it was encryption of zeros.

Note that when $w_{i,1-\mathsf{mpk}_{\mathsf{sel}}[i]}$ is encrypted by the PKE challenger, then $\mathcal{B}$ exactly simulates the view of intermediate hybrid $i-1$ to $\mathcal{A}$. Otherwise if $\mathbf{0}$ is encrypted the view is of intermediate hybrid i. Therefore, $\mathcal{A}$'s advantage in any two consecutive intermediate hybrids is negligibly close as otherwise PKE scheme is not semantically secure. Hence, using ℓ intermediate hybrids we have proved that switching $\mathsf{ct}_{i,\beta}$ from encryptions of $w_{i,\beta}$ to encryptions of $\mathbf{0}$ for $\beta \neq \mathsf{mpk}_{\mathsf{sel}}[i]$ causes at most negligible dip in $\mathcal{A}$'s advantage in Game 1. Therefore if $|\mathsf{Adv}_{\mathcal{A}}^1 - \mathsf{Adv}_{\mathcal{A}}^2|$ is non-negligible, then the PKE scheme is not semantically secure. ∎

Lemma 4.2. If GC is a secure garbling scheme, then for all PPT $\mathcal{A}$, $|\mathsf{Adv}_{\mathcal{A}}^2 - \mathsf{Adv}_{\mathcal{A}}^3| \leq \mathrm{negl}(\lambda)$ for some negligible function $\mathrm{negl}(\cdot)$.

Proof. The proof of this lemma follows from the security of our garbling scheme. First, note that the simulation based definition of garbling security can be viewed as a game based definition between a challenger and an adversary. An adversary sends a circuit $C \in \mathcal{C}_m$ and input $x \in \{0,1\}^m$. The challenger then either honestly garbles the circuit, and sends the wire keys corresponding to x, or runs the simulator to compute the garbled circuit and the wire keys for x.

Suppose there exists an adversary $\mathcal{A}$ such that $\mathsf{Adv}_{\mathcal{A}}^2 - \mathsf{Adv}_{\mathcal{A}}^3$ is non-negligible in λ. We will construct a reduction algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to break the garbling security. $\mathcal{B}$ first chooses 2ℓ public/secret key pairs and sends $\{\mathsf{pk}_{i,\beta}\}$ to the $\mathcal{A}$. $\mathcal{B}$ also chooses the base FE scheme's master public/secret keys $(\mathsf{mpk}_{\mathsf{sel}}, \mathsf{msk}_{\mathsf{sel}})$. Next, $\mathcal{A}$ sends challenge messages m_0^*, m_1^*. The reduction algorithm chooses $b \leftarrow \{0,1\}$, randomness r and computes the circuit $\mathsf{ckt} = \mathcal{C}\text{-}\mathsf{Enc}\text{-}\mathsf{pk}_{m_b^*, r}^{\ell}$. It then sends circuit ckt and input $\mathsf{mpk}_{\mathsf{sel}}$ to the garbling challenger, and receives a garbled circuit C and ℓ wire keys $\{w_i\}$. The reduction algorithm then computes $\mathsf{ct}_{i,\beta} \leftarrow \mathsf{Enc}_{\mathrm{PKE}}(\mathsf{pk}_{i,\beta}, w_i)$ if $\beta = \mathsf{mpk}[i]$, else $\mathsf{ct}_{i,\beta} \leftarrow \mathsf{Enc}_{\mathrm{PKE}}(\mathsf{pk}_{i,\beta}, \mathbf{0})$. Finally $\mathcal{B}$ sends $(C, \{\mathsf{ct}_{i,\beta}\})$ to $\mathcal{A}$ as the challenge ciphertext. The key queries are identical in both Game 2 and Game 3. Finally, the adversary sends its guess b', and if $b = b'$, the reduction algorithm guesses that ckt was honestly garbled, else it guesses that ckt and wire keys were simulated.

Note that if the garbling challenger honestly garbled circuit ckt, then $\mathcal{B}$ exactly simulates the view of Game 2 to $\mathcal{A}$. Otherwise the view is of Game 3. As a result, if $\mathsf{Adv}_{\mathcal{A}}^2 - \mathsf{Adv}_{\mathcal{A}}^3$ is non-negligible in λ, then $\mathcal{B}$ breaks the garbling scheme's security with non-negligible advantage. ∎

Lemma 4.3. If $\mathsf{FE}_{\mathrm{sel}}$ is a selectively-secure functional encryption scheme, then for all PPT $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^3 \leq \mathrm{negl}(\lambda)$ for some negligible function $\mathrm{negl}(\cdot)$.

Proof. We describe a reduction algorithm $\mathcal{B}$ which plays the selective indistinguishability based game with $\mathsf{FE}_{\mathrm{sel}}$ challenger, and simulates Game 3 for adversary $\mathcal{A}$. $\mathcal{B}$ runs the Step 1 as in Game 3, except it does not choose $\mathsf{FE}_{\mathrm{sel}}$ master public/secret key pair. It only generates 2ℓ PKE public/secret key pairs, sets $\mathsf{mpk} = \left\{\mathsf{pk}_{i,\beta}\right\}, \mathsf{msk} = \{\mathsf{sk}_{i,\beta}\}$, and sends mpk to $\mathcal{A}$. Next, $\mathcal{A}$ chooses two challenge messages (m_0^*, m_1^*),

[4] It should be noted that $\mathcal{B}$ can still answer the secret key queries during the reduction because it only needs the secret keys $\mathsf{sk}_{i,\beta}$ corresponding to the public key $\mathsf{mpk}_{\mathsf{sel}}$ (i.e. $\beta = \mathsf{mpk}_{\mathsf{sel}}[i]$). Since $\mathcal{B}$ chooses all such secret keys, therefore it can answer $\mathcal{A}$'s secret key queries.

and sends those to $\mathcal{B}$. Reduction algorithm $\mathcal{B}$ forwards (m_0^*, m_1^*) to the $\mathsf{FE}_{\mathrm{sel}}$ challenger as its challenge messages. Note that $\mathcal{B}$ is behaving as a selective adversary since it has not queried $\mathsf{FE}_{\mathrm{sel}}$ challenger for a public key before sending its challenge messages. Now $\mathsf{FE}_{\mathrm{sel}}$ challenger chooses a bit $b^* \leftarrow \{0,1\}$, runs the setup algorithm to compute key pair $(\mathsf{mpk}_{\mathsf{sel}}, \mathsf{msk}_{\mathsf{sel}})$, computes $\widetilde{\mathsf{ct}}^* \leftarrow \mathsf{Enc}_{\mathrm{sel}}(\mathsf{mpk}_{\mathsf{sel}}, m_b^*)$, and sends public key $\mathsf{mpk}_{\mathsf{sel}}$ and ciphertext $\widetilde{\mathsf{ct}}^*$ to $\mathcal{B}$. $\mathcal{B}$ receives $\mathsf{mpk}_{\mathsf{sel}}$ and $\widetilde{\mathsf{ct}}^*$ from the challenger, and it simulates the garbled circuit $(C, \{w_i\}) \leftarrow \mathsf{Sim}\left(1^\lambda, 1^\ell, 1^k, \widetilde{\mathsf{ct}}^*\right)$. Next, it computes ciphertexts $\mathsf{ct}^*_{i,\mathsf{mpk}_{\mathsf{sel}}[i]}$ as encryptions of w_i, and remaining ciphertexts as encryptions of $\mathbf{0}$. $\mathcal{B}$ sends the final challenge ciphertext ct^* as garbled circuit C and ciphertexts $\left\{\mathsf{ct}^*_{i,\beta}\right\}$ to $\mathcal{A}$. After receiving the challenge ciphertext, $\mathcal{A}$ is allowed to make polynomially many secret key queries sk_f for functions f, which $\mathcal{B}$ can answer by requesting corresponding secret keys $\mathsf{sk}_{f,\mathsf{sel}}$ from $\mathsf{FE}_{\mathrm{sel}}$ challenger, and releasing $\left\{\mathsf{sk}_{i,\mathsf{mpk}_{\mathsf{sel}}[i]}\right\}$ along with $\mathsf{sk}_{f,\mathsf{sel}}$. Finally, $\mathcal{A}$ sends its guess b' to $\mathcal{B}$, and $\mathcal{B}$ sends b' as its guess for $\mathsf{FE}_{\mathrm{sel}}$ challenger's bit b^*.

Note that $\mathcal{B}$ exactly simulates the view of Game 3 to $\mathcal{A}$. Therefore, $\mathcal{A}$'s advantage in Game 3 is negligibly small as otherwise the underlying FE scheme is not selectively-secure. Thus if $\mathsf{Adv}^3_{\mathcal{A}}$ is non-negligible, then the $\mathsf{FE}_{\mathrm{sel}}$ scheme is not selectively-secure. ■

Hence, using Lemmas 4.1, 4.2 and 4.3, we can conclude that for every PPT adversary $\mathcal{A}$ $\mathsf{Adv}^3_{\mathcal{A}} \leq \mathrm{negl}(\lambda)$, where $\mathrm{negl}(\cdot)$ is a negligible function. This completes the proof of Theorem 4.1.

5 Bundling Functionalities

In this section, we show how to transform a (standard) FE scheme to one where the public parameters can support the union of underlying message/function spaces. This transformation is similar to the one outlined in Section 4. The only difference is that instead of public key encryption, we need to use identity based encryption for encrypting the garbled circuit wire keys, and the underlying FE scheme's master public/secret keys are chosen pseudorandomly during the key generation phase.

5.1 Construction

Let $\mathsf{FE}_{\mathrm{sel}} = (\mathsf{Setup}_{\mathrm{sel}}, \mathsf{KeyGen}_{\mathrm{sel}}, \mathsf{Enc}_{\mathrm{sel}}, \mathsf{Dec}_{\mathrm{sel}})$ be a functional encryption scheme with message space $\{\mathcal{M}_n\}_n$ and function space $\{\mathcal{F}_n\}_n$, where for each $n \in \mathbb{N}$, $f \in \mathcal{F}_n$, the domain of f is $\mathcal{M}_n$. Let $\ell\text{-}\mathsf{pk}(\cdot,\cdot)$ denote the polynomial representing the size of the public key output by the setup algorithm, $\ell\text{-}\mathsf{r}_s(\cdot,\cdot)$ the randomness required by $\mathsf{Setup}_{\mathrm{sel}}$ and $\ell\text{-}\mathsf{r}_e(\cdot,\cdot)$ the randomness used by $\mathsf{Enc}_{\mathrm{sel}}$. Here, all the above polynomials take the security parameter as the first input and functionality index as the second input. For simplicity of notation, we will drop the dependence of these polynomials on the security parameter.

Tools required for our transformation : Let $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Eval})$ be a garbling scheme for circuit family $\mathcal{C} = \{\mathcal{C}_n\}_n$ such that the wire keys output by Garble have length $\ell\text{-}\mathsf{w}(\lambda)$, where λ is the security parameter. Let F be a pseudorandom function family with key space $\{\mathcal{K}_\lambda\}_\lambda$, input space $\{\{0,1\}^{2\lambda}\}_\lambda$ and output space $\{0,1\}$. Finally, we also use an identity based encryption scheme $\mathsf{IBE} = (\mathsf{Setup}_{\mathrm{IBE}}, \mathsf{Enc}_{\mathrm{IBE}}, \mathsf{KeyGen}_{\mathrm{IBE}}, \mathsf{Dec}_{\mathrm{IBE}})$ with identity space $\{\{0,1\}^{2\lambda+1}\}_\lambda$ and message space $\{\{0,1\}^{\ell\text{-}\mathsf{w}(\lambda)}\}_\lambda$.

Our transformation : We will now describe our functional encryption scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{KeyGen})$ with message space $\mathcal{M} = \cup_n\{(n,m) : m \in \mathcal{M}_n\}$ and function space $\mathcal{F} = \cup_n\{(n,f) : f \in \mathcal{F}_n\} \cup \{\epsilon\}$. Hence, each message in $\mathcal{M}$ and function in $\mathcal{F}$ has two components - the first component reveals the functionality index, and the second component is the actual message/function. For each $\mathsf{func} = (n,f) \in \mathcal{F}$ and $\mathsf{msg} = (n',m) \in \mathcal{M}$, we define $\mathsf{func}(\mathsf{msg}) = f(m)$ if $n = n'$, $\perp$ otherwise. The empty function ϵ is defined as follows: for all messages $\mathsf{msg} = (n,m) \in \mathcal{M}$, $\epsilon(\mathsf{msg}) = (n, \epsilon_n(m))$ (recall $\epsilon_n(\cdot)$ is the empty function in $\mathcal{F}_n$).

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$: The setup algorithm first runs the IBE setup to compute $(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{msk}_{\mathsf{IBE}}) \leftarrow \mathsf{Setup}_{\mathrm{IBE}}(1^\lambda)$. Next, it chooses a PRF key $K \leftarrow \mathcal{K}_\lambda$. It sets $\mathsf{mpk} = \mathsf{pp}_{\mathsf{IBE}}$ and $\mathsf{msk} = (\mathsf{msk}_{\mathsf{IBE}}, K)$.

- $\mathsf{Enc}(\mathsf{mpk}, \mathsf{msg} \in \mathcal{M}) \to \mathsf{ct}$: Let $\mathsf{msg} = (n, m)$, $t = \ell\text{-}\mathsf{pk}(n)$, and $\mathcal{C}\text{-}\mathsf{Enc}\text{-}\mathsf{pk}^t_{m,r}$ be the canonical circuit which has message m, randomness r hardwired, takes a t bit input x and computes $\mathsf{Enc}_{\mathrm{sel}}(x, m; r)$; that is, it uses the input as a public key for the base FE scheme and encrypts message m using randomness r.

  The encryption algorithm first chooses randomness $r \leftarrow \{0,1\}^{\ell\text{-}\mathsf{r}_e(n)}$. Next, it garbles the circuit $\mathcal{C}\text{-}\mathsf{Enc}\text{-}\mathsf{pk}^t_{m,r}$ by computing $(C, \{w_{i,b}\}_{i \le t, b \in \{0,1\}}) \leftarrow \mathsf{Garble}(\mathcal{C}\text{-}\mathsf{Enc}\text{-}\mathsf{pk}^t_{m,r}, 1^\lambda)$. It then encrypts the garbled wire keys by computing $\mathsf{ct}_{i,b} \leftarrow \mathsf{Enc}_{\mathrm{IBE}}(\mathsf{mpk}, w_{i,b}, (n, i, b))$. Note that both n and i can be represented as λ bit strings. The final ciphertext consists of the garbled circuit C and the $2t$ ciphertexts $\{\mathsf{ct}_{i,b}\}_{i \le t, b \in \{0,1\}}$.

- $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{func} \in \mathcal{F}) \to \mathsf{sk}_{\mathsf{func}}$: Let $\mathsf{func} = (n, f)$, $\mathsf{msk} = (\mathsf{msk}_{\mathsf{IBE}}, K)$, $s = \ell\text{-}\mathsf{r}_s(n)$ and $t = \ell\text{-}\mathsf{pk}(n)$.

  The key generation algorithm computes an s bit pseudorandom string $r = (F(K, (n, 1)), \ldots, F(K, (n, s)))$. Next, it uses r as the randomness to generate the base FE keys $(\mathsf{mpk}_n, \mathsf{msk}_n) = \mathsf{Setup}_{\mathrm{sel}}(1^\lambda, 1^n; r)$. Note that the functionality index used for generating these keys is n, and therefore the size of mpk_n is $t = \ell\text{-}\mathsf{pk}(n)$, and the amount of randomness required by $\mathsf{Setup}_{\mathrm{sel}}$ is $s = \ell\text{-}\mathsf{r}_s(n)$.

  Next, it generates IBE secret keys corresponding to the identities $(n, i, \mathsf{mpk}_n[i])$ for $i \le t$. It computes t secret keys $\mathsf{sk}_i \leftarrow \mathsf{KeyGen}_{\mathrm{IBE}}(\mathsf{msk}_{\mathsf{IBE}}, (n, i, \mathsf{mpk}_n[i]))$. Finally, it generates an FE secret key corresponding to function f by computing $\mathsf{sk}_{f,\mathsf{sel}} \leftarrow \mathsf{KeyGen}_{\mathrm{sel}}(\mathsf{msk}_n, f)$. It outputs $(\mathsf{sk}_{f,\mathsf{sel}}, \mathsf{mpk}_n, \{\mathsf{sk}_i\}_{i \le t})$ as the key for function f.

- $\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}) \to \{0, 1, \perp\}$: Let $\mathsf{sk}_f = (\mathsf{sk}_{f,\mathsf{sel}}, \mathsf{mpk}_n, \{\mathsf{sk}_i\}_{i \le t})$ and $\mathsf{ct} = (C, \{\mathsf{ct}_{i,b}\}_{i \le t, b \in \{0,1\}})$. The decryption algorithm first decrypts the appropriate garbled circuit wires. For $i \le t$, it computes $w_i = \mathsf{Dec}_{\mathrm{IBE}}(\mathsf{sk}_i, \mathsf{ct}_{i, \mathsf{mpk}_n[i]})$. It then uses these t wire keys to evaluate the garbled circuit. It computes $\widetilde{\mathsf{ct}} = \mathsf{Eval}(C, \{w_i\}_{i \le t})$. Finally, it uses the secret key $\mathsf{sk}_{f,\mathsf{sel}}$ to decrypt the ciphertext. The output is $\mathsf{Dec}_{\mathrm{sel}}(\mathsf{sk}_{f,\mathsf{sel}}, \widetilde{\mathsf{ct}})$.

Correctness : Fix any λ, message $\mathsf{msg} = (n, m) \in \mathcal{M}$, function $\mathsf{func} = (n, f) \in \mathcal{F}$ and IBE keys $(\mathsf{mpk}_{\mathsf{IBE}}, \mathsf{msk}_{\mathsf{IBE}})$. Let $(G, \{w_{i,b}\}) \leftarrow \mathsf{Garble}(\mathcal{C}\text{-}\mathsf{Enc}\text{-}\mathsf{pk}^{\ell\text{-}\mathsf{pk}(n)}_{m,r}, 1^\lambda)$ and $\mathsf{ct}_{i,b} \leftarrow \mathsf{Enc}_{\mathrm{IBE}}(\mathsf{mpk}_{\mathsf{IBE}}, w_{i,b}, (n, i, b))$. The ciphertext corresponding to message msg in our FE scheme is $(G, \{\mathsf{ct}_{i,b}\})$. Now, let us consider the key for function func. Let $(\mathsf{mpk}_n, \mathsf{msk}_n)$ be the base FE scheme's keys as computed in the key generation phase. The secret key for f in our scheme consists of IBE keys $\{\mathsf{sk}_i \leftarrow \mathsf{KeyGen}_{\mathrm{IBE}}(\mathsf{msk}_{\mathsf{IBE}}, (n, i, \mathsf{mpk}_n[i]))\}$ and $\mathsf{FE}_{\mathrm{sel}}$ key $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}_{\mathrm{sel}}(\mathsf{msk}_n, f)$.

The decryption algorithm first decrypts the IBE ciphertexts to recover the garbled circuit's wire keys $\{w_{i, \mathsf{mpk}_n[i]}\}$. Next, using $\mathsf{Eval}(G, \{w_{i, \mathsf{mpk}_n[i]}\})$, we can compute $\widetilde{\mathsf{ct}} = \mathsf{Enc}_{\mathrm{sel}}(\mathsf{mpk}_n, m; r)$. Finally, the decryption algorithm computes $\mathsf{Dec}_{\mathrm{sel}}(\mathsf{sk}_{f,\mathsf{sel}}, \widetilde{\mathsf{ct}}) = f(m)$.

5.2 Security Proof

We will now prove that the IBE scheme described above is semi-adaptive secure, as per Definition 3.3. Our proof consists of a sequence of hybrids. Let n^* denote the functionality index of the challenge inputs. The first hybrid corresponds to the semi-adaptive security game. In the second hybrid, the challenger uses a truly random function instead of a pseudorandom function. In the third hybrid, we use the security of the IBE scheme to modify the ciphertexts output as part of the challenge ciphertext. Instead of encrypting all the garbled circuit wire keys, the challenger encrypts $\mathbf{0}$ at positions that do not correspond to the base FE scheme's public key. Here, it is crucial that the challenger never outputs IBE keys corresponding to these 'off' positions. In the fourth hybrid, the garbled circuit is simulated using the challenge ciphertext of the base FE scheme. At this point, we can use the security of the base FE scheme to complete our argument.

Game 1: This is the semi-adaptive security game described in Figure 2.

1. (Setup Phase) The challenger first runs the setup algorithm by choosing $(\mathsf{mpk}_{\mathsf{IBE}}, \mathsf{msk}_{\mathsf{IBE}}) \leftarrow \mathsf{Setup}_{\mathrm{IBE}}(1^\lambda)$ and $K \leftarrow \mathcal{K}_\lambda$. It sends $\mathsf{mpk}_{\mathsf{IBE}}$ to the adversary.
2. (Challenge Phase)
   (a) $\mathcal{A}$ sends two challenge messages $\mathsf{msg}_0 = (n^*, m_0), \mathsf{msg}_1 = (n^*, m_1)$ such that $\epsilon(\mathsf{msg}_0) = \epsilon(\mathsf{msg}_1)$.
   (b) The challenger chooses a random bit $b \leftarrow \{0,1\}$, and computes the garbled circuit and its wire keys as $(C, \{w_{i,\beta}\}_{i \le t^*, \beta \in \{0,1\}}) \leftarrow \mathsf{Garble}(\mathcal{C}\text{-}\mathsf{Enc}\text{-}\mathsf{pk}^{t^*}_{m_b,r}, 1^\lambda)$, where $t^* = |\ell\text{-}\mathsf{pk}(n^*)|$.
   (c) It then encrypts the wire keys as $\mathsf{ct}_{i,\beta} \leftarrow \mathsf{Enc}_{\mathrm{IBE}}(\mathsf{mpk}_{\mathsf{IBE}}, w_{i,\beta}, (n^*, i, \beta))$.
   (d) The challenger sets $\mathsf{ct} = (C, \{\mathsf{ct}_{i,\beta}\}_{i \le t^*, \beta \in \{0,1\}})$ and sends ct to $\mathcal{A}$.
3. (Key Query Phase)
   (a) $\mathcal{A}$ queries the challenger on polynomially many functions $\mathsf{func} = (n, f) \in \mathcal{F}$ such that $\mathsf{func}(\mathsf{msg}_0) = \mathsf{func}(\mathsf{msg}_1)$. Let $s = \ell\text{-}\mathsf{r}_s(n)$, $t = \ell\text{-}\mathsf{pk}(n)$.
   (b) The challenger computes $r = (F(K,(n,1)), \ldots, F(K,(n,s)))$ and $(\mathsf{mpk}_n, \mathsf{msk}_n) = \mathsf{Setup}_{\mathrm{sel}}(1^\lambda, 1^n; r)$.
   (c) It generates the IBE secret keys as $\mathsf{sk}_i \leftarrow \mathsf{KeyGen}_{\mathrm{IBE}}(\mathsf{msk}_{\mathsf{IBE}}, (n, i, \mathsf{mpk}_n[i]))$ and base FE scheme's secret key $\mathsf{sk}_{f,\mathsf{sel}} \leftarrow \mathsf{KeyGen}_{\mathrm{sel}}(\mathsf{msk}_n, f)$.
   (d) The challenger sets $\mathsf{sk}_{\mathsf{func}} = (\mathsf{sk}_{f,\mathsf{sel}}, \mathsf{mpk}_n, \{\mathsf{sk}_i\})$ and sends $\mathsf{sk}_{\mathsf{func}}$ to $\mathcal{A}$.
4. (Guess) Finally, $\mathcal{A}$ sends its guess b' and wins if $b = b'$.

Game 2: This game is identical to the previous one, except that the challenger uses a truly random function F_{rand} instead of the pseudorandom function F.

1. (Setup Phase) The challenger first runs the setup algorithm by choosing $(\mathsf{mpk}_{\mathsf{IBE}}, \mathsf{msk}_{\mathsf{IBE}}) \leftarrow \mathsf{Setup}_{\mathrm{IBE}}(1^\lambda)$. It sends $\mathsf{mpk}_{\mathsf{IBE}}$ to the adversary.
2. (Challenge Phase)
   (a) $\mathcal{A}$ sends two challenge messages $\mathsf{msg}_0 = (n^*, m_0), \mathsf{msg}_1 = (n^*, m_1)$ such that $\epsilon(\mathsf{msg}_0) = \epsilon(\mathsf{msg}_1)$.
   (b) The challenger chooses a random bit $b \leftarrow \{0,1\}$, and computes the garbled circuit and its wire keys as $(C, \{w_{i,\beta}\}_{i \le t^*, \beta \in \{0,1\}}) \leftarrow \mathsf{Garble}(\mathcal{C}\text{-}\mathsf{Enc}\text{-}\mathsf{pk}^{t^*}_{m_b,r}, 1^\lambda)$, where $t^* = |\ell\text{-}\mathsf{pk}(n^*)|$.
   (c) It then encrypts the wire keys as $\mathsf{ct}_{i,\beta} \leftarrow \mathsf{Enc}_{\mathrm{IBE}}(\mathsf{mpk}_{\mathsf{IBE}}, w_{i,\beta}, (n^*, i, \beta))$.
   (d) The challenger sets $\mathsf{ct} = (C, \{\mathsf{ct}_{i,\beta}\}_{i \le t^*, \beta \in \{0,1\}})$ and sends ct to $\mathcal{A}$.
3. (Key Query Phase)
   (a) $\mathcal{A}$ queries the challenger on polynomially many functions $\mathsf{func} = (n, f) \in \mathcal{F}$ such that $\mathsf{func}(\mathsf{msg}_0) = \mathsf{func}(\mathsf{msg}_1)$. Let $s = \ell\text{-}\mathsf{r}_s(n)$, $t = \ell\text{-}\mathsf{pk}(n)$.
   (b) The challenger computes $r = (\ F_{\mathrm{rand}}(n,1), \ldots, F_{\mathrm{rand}}(n,s)\ )$ and $(\mathsf{mpk}_n, \mathsf{msk}_n) = \mathsf{Setup}_{\mathrm{sel}}(1^\lambda, 1^n; r)$.
   (c) It generates the IBE secret keys as $\mathsf{sk}_i \leftarrow \mathsf{KeyGen}_{\mathrm{IBE}}(\mathsf{msk}_{\mathsf{IBE}}, (n, i, \mathsf{mpk}_n[i]))$ and base FE scheme's secret key $\mathsf{sk}_{f,\mathsf{sel}} \leftarrow \mathsf{KeyGen}_{\mathrm{sel}}(\mathsf{msk}_n, f)$.
   (d) The challenger sets $\mathsf{sk}_{\mathsf{func}} = (\mathsf{sk}_{f,\mathsf{sel}}, \mathsf{mpk}_n, \{\mathsf{sk}_i\})$ and sends $\mathsf{sk}_{\mathsf{func}}$ to $\mathcal{A}$.
4. (Guess) Finally, $\mathcal{A}$ sends its guess b' and wins if $b = b'$.

Game 3: This game is identical to the previous one. Here, we are introducing some syntactical changes. In this game, the challenger chooses the base FE scheme's keys $\mathsf{mpk}_{n^*}, \mathsf{msk}_{n^*}$ immediately after receiving the challenge messages.

1. (Setup Phase) The challenger first runs the setup algorithm by choosing $(\mathsf{mpk}_{\mathsf{IBE}}, \mathsf{msk}_{\mathsf{IBE}}) \leftarrow \mathsf{Setup}_{\mathrm{IBE}}(1^\lambda)$. It sends $\mathsf{mpk}_{\mathsf{IBE}}$ to the adversary.
2. (Challenge Phase)
   (a) $\mathcal{A}$ sends two challenge messages $\mathsf{msg}_0 = (n^*, m_0), \mathsf{msg}_1 = (n^*, m_1)$ such that $\epsilon(\mathsf{msg}_0) = \epsilon(\mathsf{msg}_1)$.
   (b) The challenger chooses $(\mathsf{mpk}_{n^*}, \mathsf{msk}_{n^*}) \leftarrow \mathsf{Setup}_{\mathrm{sel}}(1^\lambda, 1^{n^*})$.
   (c) It chooses a random bit $b \leftarrow \{0,1\}$, and computes the garbled circuit and its wire keys as $(C, \{w_{i,\beta}\}_{i \le t^*, \beta \in \{0,1\}}) \leftarrow \mathsf{Garble}(\mathcal{C}\text{-}\mathsf{Enc}\text{-}\mathsf{pk}^{t^*}_{m_b,r}, 1^\lambda)$, where $t^* = |\ell\text{-}\mathsf{pk}(n^*)|$.
   (d) It then encrypts the wire keys as $\mathsf{ct}_{i,\beta} \leftarrow \mathsf{Enc}_{\mathrm{IBE}}(\mathsf{mpk}_{\mathsf{IBE}}, w_{i,\beta}, (n^*, i, \beta))$.
   (e) The challenger sets $\mathsf{ct} = (C, \{\mathsf{ct}_{i,\beta}\}_{i \le t^*, \beta \in \{0,1\}})$ and sends ct to $\mathcal{A}$.
3. (Key Query Phase)
   (a) $\mathcal{A}$ queries the challenger on polynomially many functions $\mathsf{func} = (n, f) \in \mathcal{F}$ such that $\mathsf{func}(\mathsf{msg}_0) = \mathsf{func}(\mathsf{msg}_1)$. Let $s = \ell\text{-}\mathsf{r}_s(n)$, $t = \ell\text{-}\mathsf{pk}(n)$.
   (b) The challenger chooses $(\mathsf{mpk}_n, \mathsf{msk}_n) \leftarrow \mathsf{Setup}_{\mathrm{sel}}(1^\lambda, 1^n)$ (if $\mathsf{msk}_n, \mathsf{mpk}_n$ have already been computed before, then it simply reuses those keys).
   (c) It generates the IBE secret keys as $\mathsf{sk}_i \leftarrow \mathsf{KeyGen}_{\mathrm{IBE}}(\mathsf{msk}_{\mathsf{IBE}}, (n, i, \mathsf{mpk}_n[i]))$ and base FE scheme's secret key $\mathsf{sk}_{f,\mathsf{sel}} \leftarrow \mathsf{KeyGen}_{\mathrm{sel}}(\mathsf{msk}_n, f)$.
   (d) The challenger sets $\mathsf{sk}_{\mathsf{func}} = (\mathsf{sk}_{f,\mathsf{sel}}, \mathsf{mpk}_n, \{\mathsf{sk}_i\})$ and sends $\mathsf{sk}_{\mathsf{func}}$ to $\mathcal{A}$.
4. (Guess) Finally, $\mathcal{A}$ sends its guess b' and wins if $b = b'$.

Game 4: In this game, the challenger modifies the challenge ciphertext. Instead of encrypting the garbled circuit wire keys for all $i \le t$, $\beta \in \{0,1\}$, the challenger encrypts zeroes at positions (i, β) if $\beta \neq \mathsf{mpk}_{n^*}[i]$.

1. (Setup Phase) The challenger first runs the setup algorithm by choosing $(\mathsf{mpk}_{\mathsf{IBE}}, \mathsf{msk}_{\mathsf{IBE}}) \leftarrow \mathsf{Setup}_{\mathrm{IBE}}(1^\lambda)$. It sends $\mathsf{mpk}_{\mathsf{IBE}}$ to the adversary.
2. (Challenge Phase)
   (a) $\mathcal{A}$ sends two challenge messages $\mathsf{msg}_0 = (n^*, m_0), \mathsf{msg}_1 = (n^*, m_1)$ such that $\epsilon(\mathsf{msg}_0) = \epsilon(\mathsf{msg}_1)$.
   (b) The challenger chooses $(\mathsf{mpk}_{n^*}, \mathsf{msk}_{n^*}) \leftarrow \mathsf{Setup}_{\mathrm{sel}}(1^\lambda, 1^{n^*})$.
   (c) It chooses a random bit $b \leftarrow \{0,1\}$, and computes the garbled circuit and its wire keys as $(C, \{w_{i,\beta}\}_{i \le t^*, \beta \in \{0,1\}}) \leftarrow \mathsf{Garble}(\mathcal{C}\text{-}\mathsf{Enc}\text{-}\mathsf{pk}^{t^*}_{m_b,r}, 1^\lambda)$, where $t^* = |\ell\text{-}\mathsf{pk}(n^*)|$.
   (d) It then encrypts the wire keys at half the positions, and zeroes at the remaining positions. For each i, if $\beta = \mathsf{mpk}_{n^*}[i]$, $\mathsf{ct}_{i,\beta} \leftarrow \mathsf{Enc}_{\mathrm{IBE}}(\mathsf{mpk}_{\mathsf{IBE}}, w_{i,\beta}, (n^*, i, \beta))$, else $\mathsf{ct}_{i,\beta} \leftarrow \mathsf{Enc}_{\mathrm{IBE}}(\mathsf{mpk}_{\mathsf{IBE}}, \mathbf{0}, (n^*, i, \beta))$.
   (e) It then encrypts the wire keys as $\mathsf{ct}_{i,\beta} \leftarrow \mathsf{Enc}_{\mathrm{IBE}}(\mathsf{mpk}_{\mathsf{IBE}}, w_{i,\beta}, (n^*, i, \beta))$.
   (f) The challenger sets $\mathsf{ct} = (C, \{\mathsf{ct}_{i,\beta}\}_{i \le t^*, \beta \in \{0,1\}})$ and sends ct to $\mathcal{A}$.
3. (Key Query Phase)
   (a) $\mathcal{A}$ queries the challenger on polynomially many functions $\mathsf{func} = (n, f) \in \mathcal{F}$ such that $\mathsf{func}(\mathsf{msg}_0) = \mathsf{func}(\mathsf{msg}_1)$. Let $s = \ell\text{-}\mathsf{r}_s(n)$, $t = \ell\text{-}\mathsf{pk}(n)$.
   (b) The challenger chooses $(\mathsf{mpk}_n, \mathsf{msk}_n) \leftarrow \mathsf{Setup}_{\mathrm{sel}}(1^\lambda, 1^n)$ (if $\mathsf{msk}_n, \mathsf{mpk}_n$ have already been computed before, then it simply reuses those keys).

(c) It generates the IBE secret keys as $\mathsf{sk}_i \leftarrow \mathsf{KeyGen}_{\mathrm{IBE}}(\mathsf{msk}_{\mathsf{IBE}}, (n, i, \mathsf{mpk}_n[i]))$ and base FE scheme's secret key $\mathsf{sk}_{f,\mathsf{sel}} \leftarrow \mathsf{KeyGen}_{\mathrm{sel}}(\mathsf{msk}_n, f)$.

(d) The challenger sets $\mathsf{sk}_{\mathsf{func}} = (\mathsf{sk}_{f,\mathsf{sel}}, \mathsf{mpk}_n, \{\mathsf{sk}_i\})$ and sends $\mathsf{sk}_{\mathsf{func}}$ to $\mathcal{A}$.

4. (Guess) Finally, $\mathcal{A}$ sends its guess b' and wins if $b = b'$.

Game 5: In this game, the challenger simulates the garbled circuit when computing the challenge ciphertext.

1. (Setup Phase) The challenger first runs the setup algorithm by choosing $(\mathsf{mpk}_{\mathsf{IBE}}, \mathsf{msk}_{\mathsf{IBE}}) \leftarrow \mathsf{Setup}_{\mathrm{IBE}}(1^\lambda)$. It sends $\mathsf{mpk}_{\mathsf{IBE}}$ to the adversary.

2. (Challenge Phase)

(a) $\mathcal{A}$ sends two challenge messages $\mathsf{msg}_0 = (n^*, m_0), \mathsf{msg}_1 = (n^*, m_1)$ such that $\epsilon(\mathsf{msg}_0) = \epsilon(\mathsf{msg}_1)$.

(b) The challenger chooses $(\mathsf{mpk}_{n^*}, \mathsf{msk}_{n^*}) \leftarrow \mathsf{Setup}_{\mathrm{sel}}(1^\lambda, 1^{n^*})$.

(c) It first chooses $b \leftarrow \{0,1\}$, computes $\widetilde{\mathsf{ct}} \leftarrow \mathsf{Enc}_{\mathrm{sel}}(\mathsf{mpk}_{n^*}, m_b)$.

(d) It then uses $\widetilde{\mathsf{ct}}$ to simulate the garbled circuit.
It computes $(\widetilde{C}, \{w_i\}) \leftarrow \mathsf{Sim}\left(1^\lambda, 1^{t^*}, 1^k, \widetilde{\mathsf{ct}}\right)$, where $t^* = |\ell\text{-}\mathsf{pk}(n^*)|$ and k is the size of the circuit $\mathcal{C}\text{-}\mathsf{Enc}\text{-}\mathsf{pk}^{t^*}_{m,r}$.

(e) It then encrypts the wire keys at half the positions, and zeroes at the remaining positions.
For each i, if $\beta = \mathsf{mpk}_{n^*}[i]$, $\mathsf{ct}_{i,\beta} \leftarrow \mathsf{Enc}_{\mathrm{IBE}}(\mathsf{mpk}_{\mathsf{IBE}}, w_i, (n^*, i, \beta))$, else $\mathsf{ct}_{i,\beta} \leftarrow \mathsf{Enc}_{\mathrm{IBE}}(\mathsf{mpk}_{\mathsf{IBE}}, \mathbf{0}, (n^*, i, \beta))$.

(f) The challenger sets $\mathsf{ct} = (C, \{\mathsf{ct}_{i,\beta}\}_{i \le t^*, \beta \in \{0,1\}})$ and sends ct to $\mathcal{A}$.

3. (Key Query Phase)

(a) $\mathcal{A}$ queries the challenger on polynomially many functions $\mathsf{func} = (n, f) \in \mathcal{F}$ such that $\mathsf{func}(\mathsf{msg}_0) = \mathsf{func}(\mathsf{msg}_1)$. Let $s = \ell\text{-}\mathsf{r}_s(n)$, $t = \ell\text{-}\mathsf{pk}(n)$.

(b) The challenger chooses $(\mathsf{mpk}_n, \mathsf{msk}_n) \leftarrow \mathsf{Setup}_{\mathrm{sel}}(1^\lambda, 1^n)$ (if $\mathsf{msk}_n, \mathsf{mpk}_n$ have already been computed before, then it simply reuses those keys).

(c) It generates the IBE secret keys as $\mathsf{sk}_i \leftarrow \mathsf{KeyGen}_{\mathrm{IBE}}(\mathsf{msk}_{\mathsf{IBE}}, (n, i, \mathsf{mpk}_n[i]))$ and base FE scheme's secret key $\mathsf{sk}_{f,\mathsf{sel}} \leftarrow \mathsf{KeyGen}_{\mathrm{sel}}(\mathsf{msk}_n, f)$.

(d) The challenger sets $\mathsf{sk}_{\mathsf{func}} = (\mathsf{sk}_{f,\mathsf{sel}}, \mathsf{mpk}_n, \{\mathsf{sk}_i\})$ and sends $\mathsf{sk}_{\mathsf{func}}$ to $\mathcal{A}$.

4. (Guess) Finally, $\mathcal{A}$ sends its guess b' and wins if $b = b'$.

5.2.1 Analysis

Let $\mathcal{A}$ be any PPT adversary against our construction in the semi-adaptive security game (Figure 2) and $\mathsf{Adv}^i_{\mathcal{A}}$ denote the advantage of $\mathcal{A}$ in Game i. We will show that $\mathsf{Adv}^i_{\mathcal{A}} - \mathsf{Adv}^{i+1}_{\mathcal{A}}$ is negligible in λ for all i.

Lemma 5.1. Assuming F is a secure pseudorandom function, for any PPT adversary $\mathcal{A}$, $|\mathsf{Adv}^1_{\mathcal{A}} - \mathsf{Adv}^2_{\mathcal{A}}| \le \mathrm{negl}(\lambda)$.

Proof. The proof of this lemma follows from a simple reduction to the security of PRF F. Suppose there exists an adversary $\mathcal{A}$ such that $|\mathsf{Adv}^1_{\mathcal{A}} - \mathsf{Adv}^2_{\mathcal{A}}|$ is non-negligible. We will construct an algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to break the PRF security. The reduction algorithm chooses an IBE master public/secret key pair $(\mathsf{mpk}_{\mathsf{IBE}}, \mathsf{msk}_{\mathsf{IBE}})$ and sends $\mathsf{mpk}_{\mathsf{IBE}}$ to the adversary. Next, it receives challenge messages $\mathsf{msg}_0, \mathsf{msg}_1$ with the restriction that $\epsilon(\mathsf{msg}_0) = \epsilon(\mathsf{msg}_1)$. It computes a challenge ciphertext and sends it to $\mathcal{A}$ (this step is identical in both Game 1 and Game 2). Next, the adversary queries for secret keys. For each queried function f, the reduction algorithm first computes the functionality index n and $s = \ell\text{-}\mathsf{r}_s(n)$. It then queries the PRF

challenger for PRF evaluations at inputs (n, i) for $i \leq s$. It receives string r, which it uses as randomness to compute $\mathsf{FE}_{\mathsf{sel}}$ master keys $(\mathsf{mpk}_{\mathsf{sel}}, \mathsf{msk}_{\mathsf{sel}})$. The remaining steps (computing IBE secret keys and $\mathsf{sk}_{f,\mathsf{sel}}$) are identical in both Game 1 and Game 2. It sends $\mathsf{sk}_{\mathsf{func}}$ to $\mathcal{A}$, and $\mathcal{A}$ sends its guess b'. If $b = b'$, $\mathcal{B}$ outputs 1, indicating that the oracle was a pseudorandom function, else it outputs 0, indicating that the oracle was a truly random function. Clearly, if the PRF challenger used a pseudorandom function, then $\mathcal{A}$ participates in Game 1, else it participates in Game 2. This concludes our proof.

■

Lemma 5.2. For any adversary $\mathcal{A}$, $\mathsf{Adv}^2_{\mathcal{A}} = \mathsf{Adv}^3_{\mathcal{A}}$.

Proof. The advantage of any adversary $\mathcal{A}$ is identical in Game 2 and Game 3. The only difference between the two games is that the challenger chooses $(\mathsf{mpk}_{n^*}, \mathsf{msk}_{n^*})$ immediately after receiving the challenge messages, instead of waiting for the first key query where the function is in $\mathcal{F}_{n^*}$. This does not affect the adversary's advantage. ■

Lemma 5.3. Assuming IBE is a secure identity based encryption scheme (Definition 2.3), for any PPT adversary $\mathcal{A}$, $|\mathsf{Adv}^3_{\mathcal{A}} - \mathsf{Adv}^4_{\mathcal{A}}| \leq \mathrm{negl}(\lambda)$.

Proof. Suppose there exists an adversary $\mathcal{A}$ such that $|\mathsf{Adv}^3_{\mathcal{A}} - \mathsf{Adv}^4_{\mathcal{A}}|$ is non-negligible. We will construct a reduction algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to break the security of IBE. First, $\mathcal{B}$ receives the IBE public key $\mathsf{mpk}_{\mathsf{IBE}}$, which it forwards to $\mathcal{A}$. The adversary then sends the challenge messages $\mathsf{msg}_0 = (n^*, m_0), \mathsf{msg}_1 = (n^*, m_1)$. Let $t^* = \ell\text{-}\mathsf{pk}(n^*)$. $\mathcal{B}$ chooses $(\mathsf{mpk}_{n^*}, \mathsf{msk}_{n^*}) \leftarrow \mathsf{Setup}_{\mathsf{sel}}(1^\lambda, 1^{n^*})$. It then chooses $b \leftarrow \{0, 1\}$ and computes garbled circuit C together with wire keys $\{w_{i,\beta}\}$ for message m_b. Next, it sends t^* challenge messages to the IBE challenger. For $i = 0$ to t, let $\beta'_i = 1 - \mathsf{mpk}_{n^*}[i]$. It sends challenge messages $(w_{i,\beta'_i}, \mathbf{0})$ and challenge identity (n^*, i, β'_i), and receives ciphertext ct_{i,β'_i}. The reduction algorithm constructs the remaining ciphertexts by itself and sends $(C, \{\mathsf{ct}_{i,\beta}\})$ to $\mathcal{A}$.

Next, $\mathcal{A}$ sends key queries for functions in $\mathcal{F}$. Let $\mathsf{func} = (n, f) \in \mathcal{F}$ be such a function. The reduction algorithm needs to sends IBE secret keys as part of the secret key for func. If $n \neq n^*$, then $\mathcal{B}$ can simply query the IBE challenger for secret keys. If $n = n^*$, then the reduction algorithm needs to query the IBE challenger for keys corresponding to $(n^*, i, \mathsf{mpk}_{n^*}[i])$ only. In particular, the reduction does not need to query IBE keys for the challenge identities. After receiving the IBE secret keys $\{\mathsf{sk}_i\}$, $\mathcal{B}$ computes $\mathsf{sk}_{f,\mathsf{sel}} \leftarrow \mathsf{KeyGen}_{\mathsf{sel}}(\mathsf{msk}_n, f)$ and sends $\mathsf{sk}_{\mathsf{func}} = (\mathsf{sk}_{f,\mathsf{sel}}, \mathsf{mpk}_n, \{\mathsf{sk}_i\})$ to $\mathcal{A}$. Finally, $\mathcal{A}$ sends its guess b', and $\mathcal{B}$ forwards this guess to the IBE challenger.

■

Lemma 5.4. Assuming GC is a secure garbling scheme (Definition 2.1), for any PPT adversary $\mathcal{A}$, $|\mathsf{Adv}^4_{\mathcal{A}} - \mathsf{Adv}^5_{\mathcal{A}}| \leq \mathrm{negl}(\lambda)$.

The proof of this lemma is identical to the proof of Lemma 4.2.

Lemma 5.5. Assuming $\mathsf{FE}_{\mathsf{sel}}$ is a selectively secure functional encryption scheme for function space $\{\mathcal{F}_n\}_n$ (Definition 3.2), for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^5_{\mathcal{A}} \leq \mathrm{negl}(\lambda)$.

The proof of this lemma is identical to the proof of Lemma 4.3.

References

[1] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *Proceedings of the 17th international conference on The Theory and Application of Cryptology and Information Security*, ASIACRYPT'11, 2011.

[2] Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. From selective to adaptive security in functional encryption. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 657–677, 2015.

[3] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 308–326, 2015.

[4] Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Achieving compactness generically: Indistinguishability obfuscation from non-compact functional encryption. *IACR Cryptology ePrint Archive*, 2015.

[5] Prabhanjan Vijendra Ananth and Amit Sahai. Functional encryption for turing machines. In *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part I*, pages 125–153, 2016.

[6] Nuttapong Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 557–577, 2014.

[7] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, pages 1–18, 2001.

[8] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 784–796, 2012.

[9] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 171–190, 2015.

[10] Dan Boneh and Xavier Boyen. Efficient selective-ID secure Identity-Based Encryption without random oracles. In *EUROCRYPT '04*, volume 3027 of LNCS, pages 223–238, 2004.

[11] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In *CRYPTO*, pages 443–459, 2004.

[12] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil Pairing. In *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '01, 2001.

[13] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 533–556, 2014.

[14] 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.

[15] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *CRYPTO '06*, volume 4117 of LNCS, pages 290–307, 2006.

[16] Zvika Brakerski and Vinod Vaikuntanathan. Circuit-abe from LWE: unbounded attributes and semi-adaptive security. *IACR Cryptology ePrint Archive*, 2016.

[17] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *EUROCRYPT '03*, volume 3027 of LNCS, pages 255–271, 2003.

[18] Jie Chen and Hoeteck Wee. Semi-adaptive attribute-based encryption and improved delegation for boolean formula. In *Security and Cryptography for Networks - 9th International Conference, SCN 2014, Amalfi, Italy, September 3-5, 2014. Proceedings*, pages 277–297, 2014.

[19] Clifford Cocks. An identity based encryption scheme based on Quadratic Residues. In *Cryptography and Coding, IMA International Conference*, volume 2260 of LNCS, pages 360–363, 2001.

[20] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indstinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013.

[21] Craig Gentry. Practical identity-based encryption without random oracles. In *EUROCRYPT '06*, volume 4004 of LNCS, pages 445–464, 2006.

[22] Craig Gentry and Shai Halevi. Hierarchical identity based encryption with polynomially many levels. In *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, pages 437–456, 2009.

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

[24] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In *STOC*, 2013.

[25] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, CCS '06, 2006.

[26] 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.

[27] 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.

[28] Allison B. Lewko and Brent Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 180–198, 2012.

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

[30] Adam O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010.

[31] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93, 2005.

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

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

[34] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO '84*, volume 196 of LNCS, pages 47–53, 1984.

[35] Brent Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT*, pages 114–127, 2005.

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

[37] Brent Waters. Functional encryption for regular languages. In *CRYPTO*, 2012.

[38] Brent Waters. A punctured programming approach to adaptively secure functional encryption. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 678–697, 2015.

[39] Hoeteck Wee. Dual system encryption via predicate encodings. In *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, pages 616–637, 2014.

[40] Andrew Yao. How to generate and exchange secrets. In *FOCS*, pages 162–167, 1986.