Unifying Security Notions of Functional Encryption*

Sanjam Garg† Akshayaram Srinivasan‡

Abstract

Functional Encryption (FE) generalizes the notion of traditional encryption system by providing fine-grained access to data. In a FE scheme, the holder of a secret key SK_f (associated with a function f) and a ciphertext c (encrypting plaintext x) can learn $f(x)$ but nothing else.

The indistinguishability (IND) based security notion of FE can be parameterized based on whether the adversary obtains bounded/unbounded number of challenge ciphertexts, whether she is allowed to query bounded/unbounded number of functional secret keys or whether she is forced to commit to the challenge messages prior to seeing the public parameters (selective/adaptive). It is possible to weaken further the selective security requirement (called as weakly selective setting) where the adversary is restricted to make all the function secret key queries before seeing the public parameters. These notions can be formalized as $\{xx, yy, zzz\}$-IND-FE where xx denotes the number of challenge ciphertexts, yy denotes the number of functional secret keys and zzz denotes weakly selective (Sel^*) or selective (Sel) or adaptive (Adp).

In this work, we show that *polynomially hard* $(1, 1, \mathsf{Sel}^*)$-IND-FE having *weakly compact ciphertexts* implies all other notions *generically*. Prior results required sub-exponentially hard $(1, 1, \mathsf{Sel}^*)$-IND-FE with weakly compact ciphertexts or polynomially hard $(1, \mathsf{Unb}, \mathsf{Sel})$-IND-FE to imply all other notions generically.

*Research supported in part from a DARPA/ARL SAFEWARE award, AFOSR Award FA9550-15-1-0274, and NSF CRII Award 1464397. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

†University of California, Berkeley. Email: sanjamg@berkeley.edu

‡University of California, Berkeley. Email: akshayaram@berkeley.edu

1 Introduction

Traditional encryption systems were envisioned to provide security for point-to-point communication between two parties. In a traditional encryption system, owner of the secret key can decrypt any ciphertext and completely recover the underlying plaintext whereas an eavesdropper cannot learn any information about the plaintext from the ciphertext. While this notion is sufficient for a variety of applications, the emergence of cloud computing has brought in a new set of challenges that seem to require something beyond the traditional notions. Functional Encryption [SW05, BSW11, O'N10] generalizes the notion of traditional encryption system by providing fine-grained access to data. In a Functional Encryption (FE) system holder of the master secret key MSK can derive secret keys SK_f for functions f belonging to a particular family $\mathcal{F}$. Given a ciphertext c encrypting x and the secret key SK_f, one can learn $f(x)$ but nothing else about x is leaked.

Functional encryption emerged as a generalization of Identity based encryption [Sha84, BF01, Coc01], Attribute based encryption [GPSW06], [GVW13] and Predicate Encryption [KSW08, LOS$^+$10, GVW15] each of which provided different levels of access to the underlying plaintext. Significant research on FE [KSW08, LOS$^+$10, AFV11] focused on understanding and expanding the class of functions for which FE could be realized. An alternative line of work by Sahai and Seyalioglu [SS10], Gorbunov, Vaikuntanathan and Wee [GVW12] and Goldwasser, Kalai, Popa, Vaikuntanathan and Zeldovich [GKP$^+$13] provided the first FE systems that could support all of P/poly i.e. the set of functions computable by a poly sized circuit. However, these constructions provide security only when the adversary is limited to obtaining a single function secret key.[1] Finally, works of Garg, Gentry, Halevi, Raykova, Sahai and Waters [GGH$^+$13] and Waters [Wat15] constructed FE systems supporting unbounded number of functional secret keys assuming *Indistinguishability Obfuscation* ($i\mathcal{O}$) [BGI$^+$12].

Indistinguishability based Security. The security notion of FE comes in two flavors: *simulation* based (SIM) and *indistinguishability* based (IND). While the SIM based security notion has been subject to strong lower bounds [BSW11, AGVW13], there are no such impossibility results known for IND-based security and in fact, it is known that IND-based security implies SIM based security in the random oracle model [CIJ$^+$13]. In this work, we focus on FE systems satisfying IND-based security and its refinements.

The IND-based security notion is modeled as a game between the challenger and an adversary. Informally, the task of the adversary is to distinguish between encryptions of two equal length messages x_0 and x_1 when given functional secret keys SK_f for functions f (chosen by the adversary) under the restriction that $f(x_0) = f(x_1)$.

IND-based security of FE is parameterized by two quantities: the number of challenge ciphertexts (whether the adversary is allowed to obtain a single ciphertext or a vector of ciphertexts) and the number of functional secret keys (which can either be apriori bounded or unbounded). The security notion can be further refined to selective and adaptive security. In the selective variant (Sel), the adversary is forced to commit to the challenge messages before seeing the public parameters. In the adaptive notion (Adp), the adversary is allowed to choose the challenge messages depending on the public parameters and functional secret keys for arbitrary functions of its choice.

The selective notion of security considered in literature restricts the adversary to commit to

[1] These results could be generalized to support a priori bounded number of functional secret keys.

the challenge messages before seeing the public parameters but still allows functional secret key queries to be adaptively made after seeing the challenge ciphertext and the public parameters. We consider an even weaker notion of selective security called as weakly selective security (denoted by Sel^*) that restricts the adversary to commit to her challenge messages as well as make all the function secret key queries prior to seeing the public parameters.

A central notion of efficiency of functional encryption is the size of the ciphertexts or more generally the size of the encryption circuit. There are several relaxations to the efficiency notion that have been considered in literature where the size of the ciphertext not only depends on the size of the plaintext but also on the circuit size of the functions in the function family supported by the system. A FE system is said to have *fully compact* ciphertexts (FC) if the size of the encryption circuit is some polynomial in the size of the message to be encrypted and the security parameter. A FE scheme supporting function family $\mathcal{F}$ has *weakly compact* ciphertexts (WC) if the size of the encryption circuit grows *sub-linearly* with the maximum circuit size of functions in the function family $\mathcal{F}$. A FE system is said to have *non-compact* ciphertexts (NC) if the size of the encryption circuit can depend arbitrarily on the circuit size of the functions in the function family $\mathcal{F}$.

The above parameters, the refinements and the efficiency considerations give rise to the following security and efficiency notions of IND-based FE scheme: $\{ww, xx, yyy, zz\}$-IND-FE where ww denotes the number of challenge ciphertexts, xx denotes the number of functional secret keys obtained by the adversary, yyy refers to the security setting considered i.e. Sel^* or Sel or Adp setting and zz denotes the efficiency of the scheme i.e $\{\mathsf{FC}, \mathsf{WC}, \mathsf{NC}\}$.

The focus of this work is studying the relationship between different notions of IND-based security of FE system. It can be easily seen from a standard hybrid argument that $\{1, xx, yyy, zz\}$-IND-FE implies $\{\mathsf{Unb}, xx, yyy, zz\}$-$\mathsf{IND}$-$\mathsf{FE}$. Hence in the rest of the introduction we focus on the case where the adversary obtains a single challenge ciphertext.

Prior Work. Ananth, Brakerski, Segev and Vaikuntanathan [ABSV15] gave a generic transformation from selectively secure IND-FE to adaptively secure IND-FE. This transformation preserves the number of functional secret keys i.e., starting from a selective secure scheme against unbounded functional secret keys gives an adaptively secure scheme against unbounded functional secret keys. Another observation about this reduction is that it does not preserve compactness. Even if the input to this transformation is a fully compact scheme, the resulting FE scheme is non-compact. Expressing the above results in our formulation, Ananth et al. [ABSV15] give a transformation from $(1, xx, \mathsf{Sel}, zz)$-$\mathsf{IND}$-$\mathsf{FE}$ to $(1, xx, \mathsf{Adp}, \mathsf{NC})$-$\mathsf{IND}$-$\mathsf{FE}$. Ananth and Jain [AJ15] and Bitansky and Vaikuntanathan [BV15] showed that $\{1, 1, \mathsf{Sel}^*, \mathsf{WC}\}$-$\mathsf{IND}$-$\mathsf{FE}$ implies Indistinguishability Obfuscation ($i\mathcal{O}$) [BGI+12] which is known to be powerful enough to imply $\{\mathsf{Unb}, \mathsf{Unb}, \mathsf{Adp}, \mathsf{FC}\}$-$\mathsf{IND}$-$\mathsf{FE}$ [AS16]. However, the reduction from $\{1, 1, \mathsf{Sel}^*, \mathsf{WC}\}$-$\mathsf{IND}$-$\mathsf{FE}$ to $i\mathcal{O}$ suffers an exponential loss. Hence, they had to start with a *sub-exponentially* hard $\{1, 1, \mathsf{Sel}^*, \mathsf{WC}\}$-$\mathsf{IND}$-$\mathsf{FE}$ to get $i\mathcal{O}$. Ananth, Jain and Sahai [AJS15] and Bitansky and Vaikuntanthan [BV15] gave a generic transformation from $(1, \mathsf{Unb}, \mathsf{Sel}, \mathsf{NC})$-$\mathsf{IND}$-$\mathsf{FE}$ to $(1, \mathsf{Unb}, \mathsf{Sel}, \mathsf{FC})$-$\mathsf{IND}$-$\mathsf{FE}$. This result crucially relies on the fact that the non-compact FE scheme that they start with is secure against unbounded collusions. Also, the output of this transformation is a selectively secure system irrespective of whether we start with $(1, \mathsf{Unb}, \mathsf{Adp}, \mathsf{NC})$-$\mathsf{IND}$-$\mathsf{FE}$ or $(1, \mathsf{Unb}, \mathsf{Sel}, \mathsf{NC})$-$\mathsf{IND}$-$\mathsf{FE}$.

1.1 Our Results

In this work, we show that *polynomially* hard $\{1, 1, \mathsf{Sel}^*, \mathsf{WC}\}$-IND-FE implies an *adaptive* FE scheme secure against *unbounded* collusions. In other words, to obtain (seemingly) stronger notions of security of FE namely adaptive security against unbounded collusions, it is sufficient to construct a FE scheme with the weakest possible notion of security satisfying "certain" efficiency criterion. The relationships between different security notions of IND-FE including our results is shown in Figure 1. An informal statement of our result is:

Theorem 1.1 *There exists a generic transformation from polynomially hard* $\{1, 1, \mathsf{Sel}^*, \mathsf{WC}\}$-IND-FE *to an adaptively secure* FE *with security against unbounded collusions.*

We prove the main theorem via the following two steps:

1. We give a generic transformation from $\{1, 1, \mathsf{Sel}^*, \mathsf{WC}\}$-IND-FE to $\{1, \mathsf{Unb}, \mathsf{Sel}, \mathsf{FC}\}$-IND-FE. Our reduction incurs only a *polynomial* loss in security. Interestingly, the output FE scheme is fully compact even when the starting FE scheme is only weakly compact. An additional feature of this transformation is that the resultant FE scheme is *post-challenge ciphertext* secure.[2] We view this transformation as the main contribution of this work.

2. Applying the generic transformation from selective to adaptive security of [AS16, ABSV15] we obtain a FE scheme that is adaptively secure against unbounded collusions.

Recall that the generic transformation of Ananth et al. in [ABSV15] does not preserve compactness. But as noted in the work of Hemenway et al. in [HJO+15] it is possible to combine the transformation of Ananth and Sahai in [AS16] along with adaptively secure garbled circuits [HJO+15] to obtain an adaptively secure FE scheme whose ciphertext size grows with the on-line complexity of adaptively secure garbled circuit scheme. The state of the art construction of adaptively secure garbled circuits [HJO+15] achieves an online-complexity that grows with the width of the circuit to be garbled. A FE scheme supporting function family $\mathcal{F}$ has *width compact* ciphertexts (WidC) if the size of the encryption circuit grows with the *width* of circuits in the function family $\mathcal{F}$. We obtain the following corollary:

Corollary 1.2 *There exists a generic transformation from polynomially hard* $\{1, 1, \mathsf{Sel}^*, \mathsf{WC}\}$-IND-FE *to an* $(1, \mathsf{Unb}, \mathsf{Adp}, \mathsf{WidC})$-IND-FE *scheme.*

An application of Theorem 1.1 is a construction of adaptively secure Private Linear Broadcast Encryption (PLBE) supporting apriori-bounded length identity space [BSW06, BZ14, NWZ16] with security against unbounded collusions from $\{1, 1, \mathsf{Sel}^*, \mathsf{WC}\}$-IND-FE. A private linear broadcast encryption enables generation of a ciphertext with respect to some threshold $T \in [N]$ (where $[N]$ denotes the set of identities) such that a user with identity i can decrypt the broadcast ciphertext if and only if $i \leq T$. The security guarantee is that the ciphertext "hides" the threshold T. Private Linear Broadcast encryption is used as an intermediate step to construct Traitor Tracing scheme [CFN94, BSW06]. Initial constructions of PLBE systems focused on the case where N is

[2] Post challenge ciphertext security notion is a weakening of the adaptive security where the adversary is allowed to choose the challenge messages after seeing the public parameters but before making any functional secret key queries. Post challenge ciphertext security lies in between the selective and the adaptive notions of security.

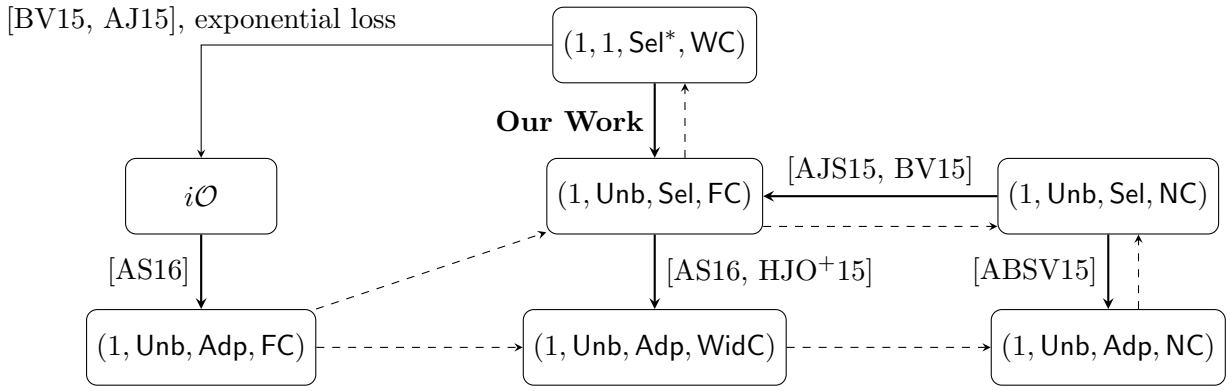

Figure 1: Relationships between different notions of IND-FE. Non-trivial relationships are given by solid arrows, and trivial relationships are given by dashed arrows.

some polynomial in the security parameter. Nishimaki, Wichs and Zhandry [NWZ16] generalized this to the case where N could be exponential by requiring that the length of the identities to be polynomial in the security parameter. They also noted (following an observation by Garg et al. in [GGH+13]) that an adaptive FE scheme with security against unbounded collusions (without any requirement on compactness) implies an adaptive PLBE that supports apriori-bounded length identities with security against unbounded collusions. Combining this result with the tracing algorithm of Nishimaki et al. [NWZ16] we get a Traitor Tracing system which supports apriori-bounded length identities from $\{1, 1, \mathsf{Sel}^*, \mathsf{WC}\}$-IND-FE.

Corollary 1.3 *There exists a construction of Traitor Tracing scheme that supports apriori bounded length identity space from* $(1, 1, \mathsf{Sel}^*, \mathsf{WC})$-IND-FE.

1.2 Our Techniques

In this section we give an overview of the techniques used in constructing selective FE with security against unbounded collusions from single-key, selective FE with weak compactness. Recall that applying the transformation of Ananth and Sahai [AS16] to the resultant scheme we can get adaptive security against unbounded collusions.

We first give a description of a selective FE scheme secure against unbounded collusions based on indistinguishability obfuscation ($i\mathcal{O}$). Though this result is not new, our construction is arguably simpler than the schemes of Garg et al. [GGH+13] and Waters [Wat15] and makes use of garbled circuits [Yao86]. Later, using techniques that are new to this work and also techniques from works of Garg et al. [GPS15, GPSZ16] we obtain a FE scheme whose security can be based on polynomially hard single-key selective FE.

1.2.1 $i\mathcal{O}$ based construction

Recall that a circuit garbling scheme (or randomized encoding in general) allows to encode an input x and a circuit C to obtain garbled input labels $\widetilde{x}$ and garbled circuit $\widetilde{C}$ respectively. Informally, the security of garbled circuits ensures that given $\widetilde{x}$ and $\widetilde{C}$ it is possible to learn $C(x)$ but nothing

Input: r
Constants: PRF key S

1. Compute $t = \mathsf{PRG}(r)$.
2. Output $(t, K = \mathsf{PRF}(S, t))$

Figure 2: Program implementing the Public Key

else. An additional feature of Yao's garbled circuits is that it is possible to encode the input x and the circuit C *separately* as long as the two encoding schemes share the same random tape.

At a high level the ciphertext of our FE scheme corresponds to garbled input labels and the functional secret key corresponds to the garbled circuit. Intuitively, from the security of gabrled circuits we can deduce that given the FE ciphertext c (encrypting x) and the functional secret key SK_f it is possible to learn $f(x)$ but nothing else. But as mentioned before, to enable encoding the input x and the circuit C separately, the random coins used must be correlated in a certain way. The main crux of the construction is in achieving this correlation using indistinguishability obfuscation ($i\mathcal{O}$).

The correlation between randomness used for garbling the input labels and the circuit is achieved by deriving the coins *pseudorandomly* using a PRF key S. This PRF key S also serves as the master secret key of our FE scheme. We now give the details of how the public key and the functional secret keys are derived from the master secret key S.

The public key of our FE scheme is an obfuscation of a program that takes as input some randomness r and outputs a "token" $t = \mathsf{PRG}(r)$ where PRG is a length doubling pseudorandom generator and a key $K = \mathsf{PRF}(S, t)$. The key K is used for deriving the input labels for the garbled circuit scheme say, that the two labels of the i-th input wire are given by $\{\mathsf{PRF}(K, i\|b)\}_{b\in\{0,1\}}$. The FE ciphertext encrypting a message m is given by the token t and the input labels corresponding to m i.e $(t, \{\mathsf{PRF}(K, i\|m_i)\}_{i\in[n]})$. The description of the program implementing the public key is given in Figure 2.

The functional secret key for a circuit C_f is an obfuscation of another program that takes as input the token t and first derives the key $K = \mathsf{PRF}(S, t)$. It then outputs a garbled circuit $\widetilde{C}_f$ where the garbled input labels are derived using key K. In particular, the input labels "encrypted" in the garbled evaluation table of $\widetilde{C}_f$ are given by $\{\mathsf{PRF}(K, i\|b)\}_{i\in[n],b\in\{0,1\}}$. The description of the program implementing the functional secret key is given in Figure 3. The FE decryption corresponds to evaluation of this garbled circuit using the input labels given in the ciphertext.

The correctness of the above construction follows from having the "correct" input labels encrypted in the garbled evaluation tables in $\widetilde{C}_f$. It remains to show that the security holds when the obfuscation is instantiated with $i\mathcal{O}$. To achieve this we use the punctured programming approach of Sahai and Waters [SW14]. We now give a high level overview of the security proof using hybrid argument.

The goal is to change from a hybrid where the adversary is given a challenge ciphertext encrypting message m_b for some $b \in \{0, 1\}$ to a hybrid where she is given a challenge ciphertext independent of the bit b. In the first hybrid we change the token t in the challenge ciphertext to an

Input: Token t
Constants: PRF key S, PRF key S_f, Circuit C_f

1. Compute $K = \mathsf{PRF}(S, t)$.
2. Compute $\mathsf{L}_{i,b_i} = \mathsf{PRF}(K, i\|b_i)$ for all $i \in [n]$ and $b_i \in \{0,1\}$.
3. Output the garbled circuit $\widetilde{C}_f$ with $\{\mathsf{L}_{i,b_i}\}_{i\in[n],b_i\in\{0,1\}}$ as the input labels and using $\mathsf{PRF}(S_f, t)$ as the random coins.

Figure 3: Program implementing the Functional Secret Key for a circuit C_f

uniformly chosen random string t^* relying on the pseudorandomness property of the PRG. Next we change the public key to be an obfuscation of a program that has the PRF key S punctured at t^* hardwired instead of S. The rest of the program is same as described in Figure 2. Intuitively, the indistinguishability follows from $i\mathcal{O}$ security as the PRG has sparse images. In the next hybrid the functional secret keys are generated as described in Figure 4 where $\widetilde{C}^*_f$ hardwired in the program is exactly equal to garbled circuit $\widetilde{C}_f$ with $\{\mathsf{PRF}(K, i\|b_i)\}_{i\in[n],b_i\in\{0,1\}}$ (where $K = \mathsf{PRF}(S, t^*)$) as the input labels and generated using $\mathsf{PRF}(S_f, t^*)$ as the random coins. The indistinguishability of the two hybrids follows from $i\mathcal{O}$ security as the two programs described in Figure 3 and Figure 4 are functionally equivalent. Now relying on the pseudorandomness at punctured point property of the PRF we change the input labels in the challenge ciphertext as well as the random coins used for generating $\widetilde{C}^*_f$ to uniformly chosen random strings. We can now change the challenge ciphertext to be independent of the bit b by relying on the security of garbled circuit. To be more precise, we change the input labels in the challenge ciphertext and $\widetilde{C}^*_f$ to be output of the garbled circuit simulator. Notice that we can still use the security of garbled circuits even if several garbled circuits share the same input labels. Thus the above construction achieves security against unbounded collusions.

1.2.2 Construction from poly hard FE

The main idea behind our construction from polynomially hard, single-key, selectively secure FE is to simulate the effect of the obfuscation in the above construction using FE. To give a better insight into our construction it would be helpful to recall the FE to $i\mathcal{O}$ transformation of Ananth and Jain [AJ15] and Bitansky and Vaikuntanathan [BV15]. Though this reduction suffers an exponential loss in security, we will be modifying this construction to achieve our goal of relying only on polynomially hard FE scheme. Parts of this section are adapted from [GPS15, GPSZ16].

FE to $i\mathcal{O}$ transformation. We describe a modification of $i\mathcal{O}$ construction from FE of Bitansky and Vaikuntanathan [BV15] (Ananth and Jain [AJ15] take a slightly different route to achieve the same result). We note that the modified construction is not sufficient to obtain $i\mathcal{O}$ security but is good enough for our purposes.

The "obfuscation" of a circuit $C : \{0,1\}^\kappa \to \{0,1\}^\kappa$ consists of the following components: a FE ciphertext CT_ϕ and $\kappa + 1$ functional secret keys $\mathsf{SK}_1, \cdots, \mathsf{SK}_{\kappa+1}$ generated using independently

Input: Token t
Constants: t^*, PRF key $S\{t^*\}$, PRF key $S_f\{t^*\}$, Circuit C_f, $\widetilde{C}^*_f$

- If $t \neq t^*$
  1. Compute $K = \mathsf{PRF}(S\{t^*\}, t)$.
  2. Compute $\mathsf{L}_{i,b_i} = \mathsf{PRF}(K, i\|b_i)$ for all $i \in [n]$ and $b_i \in \{0,1\}$.
  3. Output the garbled circuit $\widetilde{C}_f$ with $\{\mathsf{L}_{i,b_i}\}_{i\in[n],b_i\in\{0,1\}}$ as the input labels and using $\mathsf{PRF}(S_f\{t^*\}, t)$ as the random coins.
- Else, output $\widetilde{C}^*_f$.

Figure 4: Program implementing the Functional Secret Key for a circuit C_f in the Security Proof

sampled master secret keys $\mathsf{MSK}_1, \cdots, \mathsf{MSK}_{\kappa+1}$. CT_ϕ encrypts the empty string ϕ under the public key PK_1. The first κ functional secret keys $\mathsf{SK}_1, \cdots, \mathsf{SK}_\kappa$ implement the *bit-extension* functionality. To be more precise, SK_i implements the function F_i that takes as input an $(i-1)$-bit string x and outputs two ciphertexts $\mathsf{CT}_{x\|0}$ and $\mathsf{CT}_{x\|1}$ encrypting $x\|0$ and $x\|1$ respectively under PK_{i+1}. The final function secret key $\mathsf{SK}_{\kappa+1}$ implements the circuit C.

Let us discuss how to evaluate the "obfuscated" circuit on an input $x = x_1 \cdots x_\kappa$ where $x_i \in \{0,1\}$. The first step is to decrypt CT_ϕ using SK_1 to obtain $\mathsf{CT}_0, \mathsf{CT}_1$. Depending on x_1 we choose either the left encryption (CT_0) or the right encryption (CT_1) and recursively decrypt the chosen ciphertext under SK_2 and so on. After $\kappa + 1$ FE decryptions, we obtain the output of the circuit on input $x_1 \cdots x_\kappa$.

An alternate way to view this evaluation is as a traversal along a path from the root to a leaf node of a complete binary tree. The binary tree has the empty string at the root and traversal chooses either the left or the right child depending on the bits $x_1, x_2, \cdots, x_\kappa$ i.e at level i, bit x_i is used to determine whether to go left or right. We would refer to this binary tree as the *evaluation binary tree*.

Our Construction. Recall that our main idea is to simulate the effect of obfuscation by appropriately modifying the above FE to $i\mathcal{O}$ transformation. We first explain the modifications to the "obfuscation" computing the master public key.

Let $C_{pk}[S]$ (having S hardwired) be the circuit that implements the public key of our $i\mathcal{O}$-based construction. Recall that this circuit takes as input some randomness r, expands it using the PRG to obtain the token t and outputs $(t, \mathsf{PRF}(S,t))$. The goal is to produce an "obfuscation" of this circuit using FE to $i\mathcal{O}$ transformation explained above. Recall that the FE to $i\mathcal{O}$ transformation has $\kappa+1$ functional secret keys $\mathsf{SK}_1, \cdots, \mathsf{SK}_{\kappa+1}$ and an initial ciphertext CT_ϕ encrypting the empty string. The final functional secret key $\mathsf{SK}_{\kappa+1}$ implements the circuit $C_{pk}[S]$. The first observation is that we cannot naively hardwire the PRF key in the circuit C_{pk}. This is because to achieve some "meaningful" mechanisms of hiding the PRF key (via puncturing) we need to go via the $i\mathcal{O}$ route that incurs an exponential loss in security. Therefore, the first modification is to change C_{pk} such that it takes the PRF key S as input instead of having it hardwired. We also include the PRF key

S in the initial ciphertext CT_ϕ i.e CT_ϕ is now an encryption of (ϕ, S). We run into the following problem: the initial ciphertext now contains the PRF key S whereas we actually need S to be given as input to the final circuit C_{pk} that is implemented in $\mathsf{SK}_{\kappa+1}$. Therefore we need a mechanism to make the PRF key S "available" to the final functional secret key $\mathsf{SK}_{\kappa+1}$ so that it can compute PRF evaluation on the token.

On way to do this is to change every functional secret key SK_i to implement a functionality F_i that takes in (x, S) as input and outputs two ciphertexts encrypting $(x\|0, S)$ and $(x\|1, S)$ respectively. In other words, the "obfuscation" propagates S onto every root to leaf path in the evaluation binary tree. Just as in previous works [GPS15, GPSZ16] this strategy fails because the security proof needs to proceed by puncturing the PRF key S and in order to do such a puncturing we need to change the distribution of FE ciphertexts along every root to leaf path that are exponential in number. Hence this approach leads to an exponential loss in security which we wanted to avoid in first place. Therefore we need some notion of "fine-grained" puncturing of the PRF key.

To propagate the PRF key we make use of the "puncturing along the path" idea of Garg, Pandey and Srinivasan [GPS15]. This idea uses a primitive called as *prefix puncturable* PRF introduced in [GPS15]. Informally, prefix puncturable PRF allows to puncture the PRF key S at a specific prefix z to obtain S_z. The correctness guarantee is that given S_z, one can evaluate the PRF on any input x such that z is a prefix of x. The security guarantee is that as long as any adversary does not get access to S_z where z is a prefix of x, $\mathsf{PRF}(S, x)$ is indistinguishable from random string. An additional feature is that prefix puncturing can be done recursively i.e given S_z one can obtain $S_{z\|0}$ and $S_{z\|1}$. If we design a mechanism wherein the PRF key S prefix punctured at token t is available at the final functional secret key $\mathsf{SK}_{\kappa+1}$ then this can be used to derive $\mathsf{PRF}(S, t)$. Additionally, if we need to puncture the PRF key at an input x it is sufficient to change the distribution of FE ciphertexts only along the root to the leaf x in the evaluation binary tree. This gives us hope of basing security on polynomially hard FE. The second modification is to propagate the appropriate prefix punctured key instead of naively propagating the PRF key S along every path. To be more precise, every functional secret key SK_i implement a functionality F_i that takes as input x, S_x as input and outputs $\mathsf{CT}_{x\|0,S_{x\|0}}, \mathsf{CT}_{x\|1,S_{x\|1}}$ i.e it outputs ciphertext encrypting one-bit extensions along with the appropriate prefix punctured keys.

Recall that the circuit C_{pk} generates the token t as $\mathsf{PRG}(r)$ by taking r as input. If we naively try to combine this circuit with the "puncturing along the way" trick of Garg et al. we obtain S_r at the final functional secret key. It is not clear if there is a way of obtaining $S_{\mathsf{PRG}(r)}$ from S_r. Garg et al. [GPSZ16] faced a similar challenge in designing the sampler for trapdoor permutation and the solution they provide is applicable to our setting. Instead of generating the token as an output of a PRG on the input randomness r, the circuit C_{pk} takes as input P which is a public key of a semantically secure public key encryption scheme. It computes $\mathsf{PRF}(S, P)$ and outputs a public key encryption of $\mathsf{PRF}(S, P)$ using P as the public key. The token for our FE system would correspond to the public key P. We combine this circuit with the "puncturing along the way" technique of Garg et al. to obtain the "obfuscation" of our public key.

The functional secret key for a function C_f (denoted by SK_f) is constructed similarly to that of the public key. Recall that the functional secret key takes as input the token t (which is now given by the public key P) and computes $K = \mathsf{PRF}(S, t)$. It then uses the key K to derive the input garbled labels and outputs a garbled circuit $\widetilde{C}_f$. SK_f also implements the "puncturing along the way" trick of Garg et al. to obtain S_P which is used by the final circuit to derive the garbled input labels. A more detailed description of our construction is given in Section 3.

Proof Technique: "Tunneling". We now briefly explain the main proof technique which is adapted from Garg et al.'s works [GPS15, GPSZ16]. Recall that the proof of our $i\mathcal{O}$ based construction relies on the punctured programming approach of Sahai and Waters [SW14]. We also follow a similar proof strategy. Let us explain how to "puncture" the master public key on the token P. Once we have punctured the PRF key at P we can rely on the security guarantee of prefix punctured PRF to replace $\mathsf{PRF}(S, P)$ with a random string.

Puncturing the PRF key S at a string P involves "removing" S_z for every z such that z is a strict prefix of P from the "obfuscation". To get better intuition on how the puncturing works it would be helpful to view the "obfuscation" as a complete binary tree. The initial ciphertext CT_ϕ is at the root of the binary tree and each level of the binary tree is implemented by one functional secret key. The evaluation of the obfuscation corresponds to a traversal along the root to a leaf path and partial evaluation corresponds to traversal from the root to an appropriate internal node. Now puncturing the PRF key S at token P can be re-imagined as removing S_z for every z such that z is a strict prefix of P from this complete binary tree. The crucial observation that helps us to base security on polynomially hard FE is that S_z where z is a prefix of P occurs only along the path from the root to the leaf node P in this complete binary tree. Hence it is sufficient to change the distribution of the FE ciphertexts only along this path in such a manner that they don't contain S_z. To implement this change we rely on the "Hidden trapdoor mechanism" (also called as the Trojan method) of Ananth et al. in [ABSV15]. To give more details, every functional secret key SK_i implements a function F_i that has two "threads" of execution. In thread-1 or the normal mode of operation, it performs the bit-extension on input x and the prefix puncturing on input S_x. In thread-2 or the trapdoor mode, it does not perform any computation on the input (x, S_x) and instead outputs some fixed value that is hardwired. We change the FE ciphertexts in such a way that the trapdoor thread is invoked in every functional secret key when the "obfuscation" is run on input P. Metaphorically, we create a "tunnel" from the root to the leaf labeled P in the complete binary tree corresponding to the obfuscation. Additionally, we change the FE ciphertexts along the path from root to leaf P such that they do not contain any prefix punctured keys. A consequence of our "tunneling" is that along the way we would have removed S_z for every z which is a strict prefix of P from the "obfuscation."

2 Preliminaries

κ denotes the security parameter. A function $\mu(\cdot) : \mathbb{N} \to \mathbb{R}^+$ is said to be negligible if for all polynomials $\mathsf{poly}(\cdot)$, $\mu(\kappa) < \frac{1}{\mathsf{poly}(\kappa)}$ for large enough κ. For a probabilistic algorithm $\mathcal{A}$, we denote by $\mathcal{A}(x; r)$ the output of $\mathcal{A}$ on input x with the content of the random tape being r. We will omit r when it is implicit from the context. We denote $y \leftarrow \mathcal{A}(x)$ as the process of sampling y from the output distribution of $\mathcal{A}(x)$ with a uniform random tape. For a finite set S, we denote $x \leftarrow S$ as the process of sampling x uniformly from the set S. We model non-uniform adversaries $\mathcal{A} = \{\mathcal{A}_\kappa\}$ as circuits such that for all κ, $\mathcal{A}_\kappa$ is of size $p(\kappa)$ where $p(\cdot)$ is a polynomial. We will drop the subscript κ from the adversary's description when it is clear from the context. We will also assume that all algorithms are given the unary representation of security parameter 1^κ as input and will not mention this explicitly when it is clear from the context. We will use PPT to denote Probabilistic Polynomial Time algorithm. We denote $[\kappa]$ to be the set $\{1, \cdots, k\}$. We will use $\mathsf{negl}(\cdot)$ to denote an unspecified negligible function and $\mathsf{poly}(\cdot)$ to denote an unspecified polynomial. We assume without loss of generality that all cryptographic randomized algorithms use κ-bits of randomness.

If the algorithm needs more than κ-bit of randomness it can extend to arbitrary polynomial stretch using a pseudorandom generator (PRG).

A binary string $x \in \{0,1\}^\kappa$ is represented as $x_1 \cdots x_\kappa$. x_1 is the most significant (or the highest order bit) and x_κ is the least significant (or the lowest order bit). The i-bit prefix $x_1 \cdots x_i$ of the binary string x is denoted by $x_{[i]}$. We use $x \| y$ to denote concatenation of binary strings x and y. We say that a binary string y is a prefix of x if and only if there exists a string $z \in \{0,1\}^*$ such that $x = y \| z$.

Puncturable pseudorandom Function. We recall the notion of puncturable pseudorandom function from [SW14]. The construction of pseudorandom function given in [GGM86] satisfies the following definition [BW13, KPTZ13, BGI14].

Definition 2.1 *A puncturable pseudorandom function* PRF *is a tuple of PPT algorithms* $(\mathsf{KeyGen}_{\mathsf{PRF}}, \mathsf{PRF}, \mathsf{Punc})$ *with the following properties:*

- ***Efficiently Computable:*** *For all* κ *and for all* $S \leftarrow \mathsf{KeyGen}_{\mathsf{PRF}}(1^\kappa)$, $\mathsf{PRF}_S : \{0,1\}^{\mathsf{poly}(\kappa)} \rightarrow \{0,1\}^\kappa$ *is polynomial time computable.*
- ***Functionality is preserved under puncturing:*** *For all* κ, *for all* $y \in \{0,1\}^\kappa$ *and* $\forall x \neq y$,

$$\Pr[\mathsf{PRF}_{S\{y\}}(x) = \mathsf{PRF}_S(x)] = 1$$

  where $S \leftarrow \mathsf{KeyGen}_{\mathsf{PRF}}(1^\kappa)$ *and* $S\{y\} \leftarrow \mathsf{Punc}(S, y)$.
- ***Pseudorandomness at punctured points:*** *For all* κ, *for all* $y \in \{0,1\}^\kappa$, *and for all poly sized adversaries* $\mathcal{A}$

$$|\Pr[\mathcal{A}(\mathsf{PRF}_S(y), S\{y\}) = 1] - \Pr[\mathcal{A}(U_\kappa, S\{y\}) = 1]| \leq \mathsf{negl}(\kappa)$$

  where $S \leftarrow \mathsf{KeyGen}_{\mathsf{PRF}}(1^\kappa)$, $S\{y\} \leftarrow \mathsf{Punc}(S, y)$ *and* U_κ *denotes the uniform distribution over* $\{0,1\}^\kappa$.

Symmetric Key Encryption. A Symmetric-Key Encryption scheme SKE is a tuple of algorithms $(\mathsf{SK.KeyGen}, \mathsf{SK.Enc}, \mathsf{SK.Dec})$ with the following syntax:

- $\mathsf{SK.KeyGen}(1^\kappa)$: Takes as input an unary encoding of the security parameter κ and outputs a symmetric key SK.
- $\mathsf{SK.Enc}_{SK}(m)$: Takes as input a message $m \in \{0,1\}^*$ and outputs an encryption C of the message m under the symmetric key SK.
- $\mathsf{SK.Dec}_{SK}(C)$: Takes as input a ciphertext C and outputs a message m'.

We say that SKE is *correct* if for all κ and for all messages $m \in \{0,1\}^*$, $\Pr[\mathsf{SK.Dec}_{SK}(C) = m] = 1$ where $SK \leftarrow \mathsf{SK.KeyGen}(1^\kappa)$ and $C \leftarrow \mathsf{SK.Enc}_{SK}(m)$.

Definition 2.2 *For all* κ *and for all polysized adversaries* $\mathcal{A}$,

$$\left|\Pr[\mathsf{Expt}_{1^\kappa,0,\mathcal{A}} = 1] - \Pr[\mathsf{Expt}_{1^\kappa,1,\mathcal{A}} = 1]\right| \leq \mathsf{negl}(\kappa)$$

where $\mathsf{Expt}_{1^\kappa,b,\mathcal{A}}$ *is defined below:*

- ***Challenge Message Queries:*** *The adversary $\mathcal{A}$ outputs two messages m_0 and m_1 such that $|m_0| = |m_1|$ for all $i \in [n]$.*
- *The challenger samples $SK \leftarrow \mathsf{SK.KeyGen}(1^\kappa)$ and generates the challenge ciphertext C where $C \leftarrow \mathsf{SK.Enc}_{SK}(m_b)$. It then sends C to $\mathcal{A}$.*
- *Output is b' which is the output of $\mathcal{A}$.*

Remark 2.3 *We will denote range of a secret key* SK *(denoted by* $\mathsf{Range}_n(SK)$*) to be* $\{\mathsf{SK.Enc}(SK, x)\}_{x\in\{0,1\}^n}$ *for a specific n. We will require that for any two secret keys SK_1, SK_2 where $SK_1 \neq SK_2$ we have* $\mathsf{Range}_n(SK_1) \cap \mathsf{Range}_n(SK_2) = \phi$ *with overwhelming probability. We will also require that the existence of an efficient procedure that checks if a given ciphertext c belongs to* $\mathsf{Range}_n(SK)$ *for a particular secret key SK.*

Public Key Encryption. A public-key Encryption scheme PKE is a tuple of algorithms ($\mathsf{PK.KeyGen}$, $\mathsf{PK.Enc}, \mathsf{PK.Dec}$) with the following syntax:

- $\mathsf{PK.KeyGen}(1^\kappa)$: Takes as input an unary encoding of the security parameter κ and outputs a public key, secret key pair (pk, sk).
- $\mathsf{PK.Enc}_{pk}(m)$: Takes as input a message $m \in \{0,1\}^*$ and outputs an encryption C of the message m under the public key pk.
- $\mathsf{PK.Dec}_{sk}(C)$: Takes as input a ciphertext C and outputs a message m'.

We say that PKE is *correct* if for all κ and for all messages $m \in \{0,1\}^*$, $\Pr[\mathsf{PK.Dec}_{sk}(C) = m] = 1$ where $(pk, sk) \leftarrow \mathsf{PK.KeyGen}(1^\kappa)$ and $C \leftarrow \mathsf{PK.Enc}_{pk}(m)$.

Definition 2.4 *For all κ and for all polysized adversaries $\mathcal{A}$ and for all messages $m_0, m_1 \in \{0,1\}^*$ such that $|m_0| = |m_1|$,*

$$|\Pr[\mathcal{A}(pk, \mathsf{PK.Enc}_{pk}(m_0)) = 1] - \Pr[\mathcal{A}(pk, \mathsf{PK.Enc}_{pk}(m_1)) = 1]| \leq \mathsf{negl}(\kappa)$$

where $(pk, sk) \leftarrow \mathsf{PK.KeyGen}(1^\kappa)$.

Prefix Puncturable pseudorandom Functions. We now define the notion of prefix puncturable pseudorandom function PPRF which is satisfied by the construction of the pseudorandom function in [GGM86].

Definition 2.5 *A prefix puncturable pseudorandom function* PPRF *is a tuple of PPT algorithms* ($\mathsf{KeyGen}_{\mathsf{PPRF}}, \mathsf{PrefixPunc}$) *satisfying the following properties:*

- ***Functionality is preserved under* repeated *puncturing:*** *For all κ, for all $y \in \cup_{k=0}^{\mathsf{poly}(\kappa)}\{0,1\}^k$ and for all $x \in \{0,1\}^{\mathsf{poly}(\kappa)}$ such that there exists a $z \in \{0,1\}^*$ s.t. $x = y\|z$,*

$$\Pr[\mathsf{PrefixPunc}(\mathsf{PrefixPunc}(S, y), z) = \mathsf{PrefixPunc}(S, x)] = 1$$

where $S \leftarrow \mathsf{KeyGen}_{\mathsf{PPRF}}$.

- ***Pseudorandomness at punctured prefix:*** *For all κ, for all $x \in \{0,1\}^{\mathsf{poly}(\kappa)}$, and for all poly sized adversaries $\mathcal{A}$*

$$|\Pr[\mathcal{A}(\mathsf{PrefixPunc}(S, x), \mathsf{Keys}) = 1] - \Pr[\mathcal{A}(U_\kappa, \mathsf{Keys}) = 1]| \leq \mathsf{negl}(\kappa)$$

where $S \leftarrow \mathsf{KeyGen}_{\mathsf{PRF}}(1^\kappa)$ and $\mathsf{Keys} = \{\mathsf{PrefixPunc}(S, x_{[i-1]}\|(1 - x_i))\}_{i\in[\mathsf{poly}(\kappa)]}$.

Notation. For brevity of notation, we will be denoting $\mathsf{PrefixPunc}(S, y)$ by S_y.

Garbled Circuits. We now define the circuit garbling scheme of Yao [Yao86] and state the required properties.

Definition 2.6 *A circuit garbling scheme is a tuple of PPT algorithms given by* $(\mathsf{Garb.Circuit}, \mathsf{Garb.Eval})$ *with the following syntax:*

- $\mathsf{Garb.Circuit}(C)$: *This is a randomized algorithm that takes in the circuit to be garbled and outputs garbled circuit and the set of garbled input labels:* $\widetilde{C}, \{\mathsf{Inp}_{i,b_i}\}_{i\in[\kappa],b_i\in\{0,1\}}$.
- $\mathsf{Garb.Eval}(\widetilde{C}, \{\mathsf{Inp}_{i,x_i}\}_{i\in[\kappa]})$: *This is a deterministic algorithm that takes in* $\{\mathsf{Inp}_{i,x_i}\}_{i\in[\kappa]}$ *and* $\widetilde{C}$ *as input and outputs a string* y.

Definition 2.7 (Correctness) *We say a circuit garbling scheme is correct if for all circuits* C *and for all inputs* x*:*

$$\Pr[\mathsf{Garb.Eval}(\widetilde{C}, \{\mathsf{Inp}_{i,x_i}\}_{i\in[\kappa]}) = C(x)] = 1$$

where $\widetilde{C}, \{\mathsf{Inp}_{i,b_i}\}_{i\in[\kappa],b_i\in[\kappa]} \leftarrow \mathsf{Garb.Circuit}(K, C)$.

Definition 2.8 (Security) *There exists a simulator* Sim *such that for all circuits* C *and input* x*:*

$$\{\widetilde{C}, \{\mathsf{Inp}_{i,x_i}\}_{i\in[\kappa]}\} \stackrel{c}{\approx} \{\mathsf{Sim}(1^\kappa, C, C(x))\}$$

Lemma 2.9 ([Yao86],[LP09]) *Assuming the existence of one-way functions there exists a circuit garbling scheme satisfying the security notion given in Definition 2.8.*

2.1 Functional Encryption

We recall the syntax and security notions of functional encryption [BSW11, O'N10].

A functional encryption FE with the message space $\{0,1\}^*$ and function space $\mathcal{F}$ is a tuple of PPT algorithms $(\mathsf{FE.Setup}, \mathsf{FE.Enc}, \mathsf{FE.KeyGen}, \mathsf{FE.Dec})$ having the following syntax:

- $\mathsf{FE.Setup}(1^\kappa)$: Takes as input the unary encoding of the security parameter κ and outputs a public key PK and a master secret key MSK.
- $\mathsf{FE.Enc}(\mathsf{PK}, m)$: Takes as input a message $m \in \{0,1\}^*$ and outputs an encryption c of m under the public key PK.
- $\mathsf{FE.KeyGen}(\mathsf{MSK}, f)$: Takes as input the master secret key MSK and a function $f \in \mathcal{F}$ (given as a circuit) as input and outputs the function key SK_f.
- $\mathsf{FE.Dec}(\mathsf{SK}_f, c)$: Takes as input the function key SK_f and the ciphertext c and outputs a string y.

Definition 2.10 (Correctness) *The functional encryption scheme* FE *is correct if for all* κ *and for all messages* $m \in \{0,1\}^*$*,*

$$\Pr\left[y = f(m) \middle| \begin{array}{l} (\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{FE.Setup}(1^\kappa) \\ c \leftarrow \mathsf{FE.Enc}(\mathsf{PK}, m) \\ \mathsf{SK}_f \leftarrow \mathsf{FE.KeyGen}(\mathsf{MSK}, f) \\ y \leftarrow \mathsf{FE.Dec}(\mathsf{SK}_f, c) \end{array} \right] = 1$$

In the introduction we took a joint view of the security and efficiency of a functional encryption scheme. We now decouple the efficiency and the security requirement. We first describe the security requirements and then describe the efficiency requirements.

Security. The indistinguishability based security of function encryption scheme is parameterized by two quantities: the number of challenge ciphertexts (which is apriori bounded or unbounded) and the number of functional secret keys (which is either apriori bounded or unbounded). The security notion can be further refined to: selective and adaptive security. In the selective variant, the adversary is forced to commit to the challenge messages prior to seeing the public parameters. In the adaptive notion, the adversary can choose the challenge messages depending on the public parameters and functional secret keys for arbitrary functions of its choice. We consider a further weakening of the selective security notion denoted as weakly selective security where the adversary is forced to commit to the functional secret keys before viewing the public parameters.

We now give the formal definitions of the security notions. We start with the weakest notion of security namely *weakly selective security.*

Definition 2.11 (Weakly Selective Security) *The functional encryption scheme is said to be* $(\mathsf{Unb}, \mathsf{Unb}, \mathsf{Sel}^*)$-$\mathsf{IND}$ *secure if for all* κ *and for all poly sized adversaries* $\mathcal{A}$*,*

$$\left|\Pr[\mathsf{Expt}_{\mathsf{Sel}^*,1^\kappa,0,\mathcal{A}} = 1] - \Pr[\mathsf{Expt}_{\mathsf{Sel}^*,1^\kappa,1,\mathcal{A}} = 1]\right| \leq \mathsf{negl}(\kappa)$$

where $\mathsf{Expt}_{\mathsf{Sel},1^\kappa,b,\mathcal{A}}$ *is defined below:*

- ***Challenge Message Queries:*** *The adversary* $\mathcal{A}$ *outputs two message vectors* $\vec{m}_0$*,* $\vec{m}_1$ *such that* $|\vec{m}_0| = |\vec{m}_1|$ *and for all* $i \in [|\vec{m}_0|], |m_{0,i}| = |m_{1,i}|$ *and a set of functions* $f_1, \cdots, f_q \in \mathcal{F}$ *to the challenger. The parameter* q *and the size of message vectors are apriori-unbounded.*
- *The challenger samples* $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{FE.Setup}(1^\kappa)$ *and generates the challenge ciphertext vector* $\vec{c} \leftarrow \mathsf{FE.Enc}(\mathsf{PK}, \vec{m}_b)$*. The challenger also computes* $\mathsf{SK}_{f_i} \leftarrow \mathsf{FE.KeyGen}(\mathsf{MSK}, f_i)$ *for all* $i \in [q]$*. It then sends* $(\mathsf{PK}, \vec{c}), \{\mathsf{SK}_{f_i}\}_{i\in[q]}$ *to* $\mathcal{A}$*.*
- *If* $\mathcal{A}$ *makes a query* f_j *for some* $j \in [q]$ *to functional key generation oracle such that for any* $i \in [|\vec{m}_0|]$*,* $f_j(m_{0,i}) \neq f_j(m_{1,i})$*, output of the experiment is* $\perp$*. Otherwise, the output is* b' *which is the output of* $\mathcal{A}$*.*

Remark 2.12 *We say that the functional encryption scheme* FE *is* ***single-key, weakly selectively secure*** *if the adversary* $\mathcal{A}$ *in* $\mathsf{Expt}_{\mathsf{Sel}^*,1^\kappa,b,\mathcal{A}}$ *is allowed to obtain the functional key for a single function* f*.*

We now give the definition of selectively secure FE.

Definition 2.13 (Selective Security) *The functional encryption scheme is said to be* $(\mathsf{Unb}, \mathsf{Unb}, \mathsf{Sel})$-$\mathsf{IND}$ *secure if for all* κ *and for all poly sized adversaries* $\mathcal{A}$*,*

$$\left|\Pr[\mathsf{Expt}_{\mathsf{Sel},1^\kappa,0,\mathcal{A}} = 1] - \Pr[\mathsf{Expt}_{\mathsf{Sel},1^\kappa,1,\mathcal{A}} = 1]\right| \leq \mathsf{negl}(\kappa)$$

where $\mathsf{Expt}_{\mathsf{Sel},1^\kappa,b,\mathcal{A}}$ *is defined below:*

- ***Challenge Message Queries:*** *The adversary $\mathcal{A}$ outputs two message vectors $\vec{m}_0$, $\vec{m}_1$ such that $|\vec{m}_0| = |\vec{m}_1|$ and for all $i \in [|\vec{m}_0|], |m_{0,i}| = |m_{1,i}|$ to the challenger.*
- *The challenger samples $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{FE.Setup}(1^\kappa)$ and generates the challenge ciphertext vector $\vec{c} \leftarrow \mathsf{FE.Enc}(\mathsf{PK}, \vec{m}_b)$. It then sends $(\mathsf{PK}, \vec{c})$ to $\mathcal{A}$.*
- ***Function Queries****: $\mathcal{A}$ adaptively chooses a function $f \in \mathcal{F}$ and sends it to the challenger. The challenger responds with $\mathsf{SK}_f \leftarrow \mathsf{FE.KeyGen}(\mathsf{MSK}, f)$. The number of function queries made by the adversary is unbounded.*
- *If $\mathcal{A}$ makes a query f to functional key generation oracle such that for any $i \in [|\vec{m}_0|]$, $f(m_{0,i}) \neq f(m_{1,i})$, output of the experiment is $\perp$. Otherwise, the output is b' which is the output of $\mathcal{A}$.*

We now give the definition of adaptively secure FE.

Definition 2.14 (Adaptive Security) *The functional encryption scheme is said to be* $(\mathsf{Unb}, \mathsf{Unb}, \mathsf{Adp})$-$\mathsf{IND}$ *secure if for all κ and for all poly sized adversaries $\mathcal{A}$,*

$$\left|\Pr[\mathsf{Expt}_{\mathsf{Adp},1^\kappa,0,\mathcal{A}} = 1] - \Pr[\mathsf{Expt}_{\mathsf{Adp},1^\kappa,1,\mathcal{A}} = 1]\right| \leq \mathsf{negl}(\kappa)$$

where $\mathsf{Expt}_{\mathsf{Adp},1^\kappa,b,\mathcal{A}}$ is defined below:

- *The challenger samples $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{FE.Setup}(1^\kappa)$ and PK to the adversary.*
- ***Phase-1:*** *$\mathcal{A}$ adaptively chooses a function $f \in \mathcal{F}$ and sends it to the challenger. The challenger responds with $\mathsf{SK}_f \leftarrow \mathsf{FE.KeyGen}(\mathsf{MSK}, f)$.*
- ***Challenge Message Queries:*** *The adversary $\mathcal{A}$ outputs two message vectors $\vec{m}_0$, $\vec{m}_1$ such that $|\vec{m}_0| = |\vec{m}_1|$ and for all $i \in [|\vec{m}_0|], |m_{0,i}| = |m_{1,i}|$ to the challenger.*
- ***Phase-2****: $\mathcal{A}$ adaptively chooses a function $f \in \mathcal{F}$ and sends it to the challenger. The challenger responds with $\mathsf{SK}_f \leftarrow \mathsf{FE.KeyGen}(\mathsf{MSK}, f)$.*
- *If $\mathcal{A}$ makes a query f to functional key generation oracle (in Phase-1 or Phase-2) such that for any $i \in [|\vec{m}_0|]$, $f(m_{0,i}) \neq f(m_{1,i})$, output of the experiment is $\perp$. Otherwise, the output is b' which is the output of $\mathcal{A}$.*

It can be easily shown through a standard hybrid argument that $(1, yy, zzz)$-IND based security notion implies (Unb, yy, zzz)-IND based security. Thus, without loss of generality we would consider $(1, yy, zzz)$-IND based security.

Lemma 2.15 *There exists a generic transformation from a* FE*-scheme satisfying* $(1, yy, zzz)$-IND *based security to one that satisfies* (Unb, yy, zzz)-IND *based security.*

Efficiency We now define the efficiency requirements of a FE scheme.

Definition 2.16 (Fully Compact) *A functional encryption scheme* FE *is said to be fully compact if for all* $\kappa \in \mathbb{N}$ *and for all* $m \in \{0,1\}^*$ *the running time of the encryption algorithm* $\mathsf{FE.Enc}$ *is* $\mathsf{poly}(\kappa, |m|)$.

Definition 2.17 (Weakly Compact) *A functional encryption scheme is said to be weakly compact if the running time of the encryption algorithm* $\mathsf{FE.Enc}$ *is* $|\mathcal{F}|^{1-\varepsilon}.\mathsf{poly}(\kappa, |m|)$ *for some* $\varepsilon > 0$ *where* $|\mathcal{F}| = \max_{f \in \mathcal{F}} |C_f|$ *where* C_f *is the circuit implementing* f.

A functional encryption scheme is said to have *non-compact ciphertexts* if the running time of the encryption algorithm can depend arbitrarily on the maximum circuit size of the function family.

3 Our Transformation

In this section we describe our transformation from single-key, selectively secure weakly compact functional encryption to selective functional encryption scheme with fully compact ciphertexts that is secure against unbounded key generation queries. We will assume that single-key scheme is fully compact and later relax the requirement in Section 4.

The primitives that are used in the transformation are:

- A $(1, 1, \mathsf{Sel}^*, \mathsf{FC})$-$\mathsf{IND\text{-}FE}$ scheme $(\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$.
- A prefix puncturable PRF $(\mathsf{PPRF}, \mathsf{PrefixPunc})$.
- A Circuit garbling scheme $(\mathsf{Garb.Circuit}, \mathsf{Garb.Eval})$.
- A public key encryption scheme $(\mathsf{PK.KeyGen}, \mathsf{PK.Enc}, \mathsf{PK.Dec})$.
- A symmetric key encryption scheme $(\mathsf{SK.KeyGen}, \mathsf{SK.Enc}, \mathsf{SK.Dec})$.

The output of the transformation is a FE scheme $(\mathsf{UFE.Setup}, \mathsf{UFE.KeyGen}, \mathsf{UFE.Enc}, \mathsf{UFE.Dec})$ that is secure against unbounded key generation queries. The formal description our construction appears in Figure 6 and we provide an overview below.

Overview. The master public key of our UFE scheme consists of $\kappa + 1$ function secret keys $\mathsf{SK}^1_1, \cdots, \mathsf{SK}^1_{\kappa+1}$ (generated using independently sampled $\{\mathsf{PK}^1_i, \mathsf{MSK}^1_i\}_{i \in [\kappa+1]}$) and an initial FE ciphertext CT^1_ϕ. The initial ciphertext CT^1_ϕ encrypts (ϕ, S) under the public key PK^1_1 where S is a fresh prefix puncturable key working on κ-bit inputs. The master secret key is set to S. We now describe the functionality implemented by the $\kappa + 1$ function secret keys given by $\mathsf{SK}^1_1, \cdots, \mathsf{SK}^1_{\kappa+1}$.

The first κ keys $\mathsf{SK}^1_1, \cdots, \mathsf{SK}^1_\kappa$ implement a function that performs *bit-extension* and *prefix puncturing*. To be more precise, SK^1_i corresponds to a function BitExt^1_i that takes in $(y \in \{0,1\}^{i-1}, S_y)$ where S_y is a PRF key S prefix punctured at y as input and outputs FE encryptions of $(y\|0, S_{y\|0})$ and $(y\|1, S_{y\|1})$ under PK^1_{i+1}. [3] The final key $\mathsf{SK}^1_{\kappa+1}$ corresponds to the output functionality that on input (x, S_x), outputs an encryption of S_x using x as the public key.[4]

[3] In this exposition we ignore the randomness required for computing the encryptions. In the actual construction the randomness is generated using an independent prefix puncturable PRF key.

[4] We again use a separate prefix puncturable PRF key to generate the randomness needed for encryption.

To encrypt a message m, we first sample a public key-secret key pair (pk, sk). We then decrypt the initial ciphertext CT^1_ϕ (which is part of the master public key) using SK^1_1 to obtain an FE encryption of $(0, S_0)$ and $(1, S_1)$ under PK^1_2. Depending on the first bit of pk we choose either the left encryption or the right and then repeat the decryption under SK^1_2 and so on. We call this procedure of successively decrypting along the FE-binary tree as *Iterated Decryption* on a particular input. Iterated decryption procedure is illustrated in Figure 5. In this case, we iteratively decrypt the master public key on input pk. Thus, after $\kappa+1$ FE decryptions we get a public key encryption of S_{pk} under pk. We then recover S_{pk} using the secret key sk. The UFE ciphertext is given by $(pk, \{\mathsf{PRF}(S_{pk}, i\|m_i)\}_{i\in[\kappa]})$. In the introduction, we had remarked that $\{\mathsf{PRF}(S_{pk}, i\|m_i)\}_{i\in[\kappa]}$ would serve as the input labels of the a garbled circuit. In the actual construction, we would be deviating from this intuition as we would be using $\{\mathsf{PRF}(S_{pk}, i\|m_i)\}_{i\in[\kappa]}$ as a key to decrypt encrypted garbled input labels.

The structure of the secret key SK_f corresponding to a function f is similar to that of the master public key in the sense that it too consists of $\kappa+1$ function secret keys $\mathsf{SK}^f_1, \cdots, \mathsf{SK}^f_{\kappa+1}$ and an initial ciphertext CT^f_ϕ encrypting the empty string ϕ and the master secret key S. Looking ahead, our UFE decryption procedure on a ciphertext $(pk, \{\{\mathsf{PRF}(S_{pk}, i\|m_i)\}_{i\in[\kappa]})$ and a secret key SK_f corresponds to the iterated decryption of the SK_f with the pk as input and a subsequent evaluation of a garbled circuit (more details follow). The first κ secret keys $\mathsf{SK}^f_1, \cdots, \mathsf{SK}^f_\kappa$ are exactly same as that of the master public key i.e they implement the bit-extension and the prefix puncturing functionality. The last functionality takes in S_{pk} as input, garbles C_f where C_f is the circuit computing f, encrypts the input labels under $\mathsf{PRF}(S_{pk}, i\|b_i)$ for all $i \in [\kappa]$ and $b_i \in \{0,1\}$ and outputs the garbled circuit along with the encrypted input labels.[5] The decryption procedure of our scheme obtains the garbled circuit and the encrypted input labels by iterated decryption. We then decrypt the relevant ciphertexts using $\mathsf{PRF}(S_{pk}, i\|m_i)$ to obtain the corresponding input labels. The output is the evaluation of the garbled circuit on the input labels.

Correctness. We first argue that our FE ciphertext is distributed as $(pk, \{\mathsf{PRF}(S_{pk}, i\|m_i)\}_{i\in[\kappa]})$. From the correctness of the iterated decryption procedure of the master public key with input pk (which in turn depends on the correctness of the FE decryption under $\mathsf{SK}^1_1, \cdots, \mathsf{SK}^1_{\kappa+1}$) we observe that decrypting CT_{pk} under $\mathsf{SK}^1_{\kappa+1}$ yields a public key encryption of S_{pk} under public key pk. From the correctness of public key decryption, we correctly recover S_{pk}. Hence our FE ciphertext is distributed as $(pk, \{\mathsf{PRF}(S_{pk}, i\|m_i)\}_{i\in[\kappa]})$.

Now from the correctness of iterated decryption procedure of SK_f with pk as input (which in turn depends on the correctness of the FE decryption under $\mathsf{SK}^f_1, \cdots, \mathsf{SK}^1_{\kappa+1}$), we obtain $\widetilde{C_f}, c_{i,b_i}$ where $c_{i,b_i} \leftarrow \mathsf{SK.Enc}(\mathsf{PRF}(i\|m_i), \mathsf{Inp}_{i,b_i})$. It follows from the correctness of $\mathsf{SK.Dec}$ we correctly obtain $\{\mathsf{Inp}_{i,m_i}\}_{i\in[\kappa]}$. The correctness of our UFE decryption follows from the correctness of garbled circuit evaluation.

3.1 Security

In this section we prove that our UFE construction described in Figure 6 is selectively secure against unbounded collusions.

[5] The randomness required for garbling and encrypting is drawn pseudorandomly using a separate PRF key.

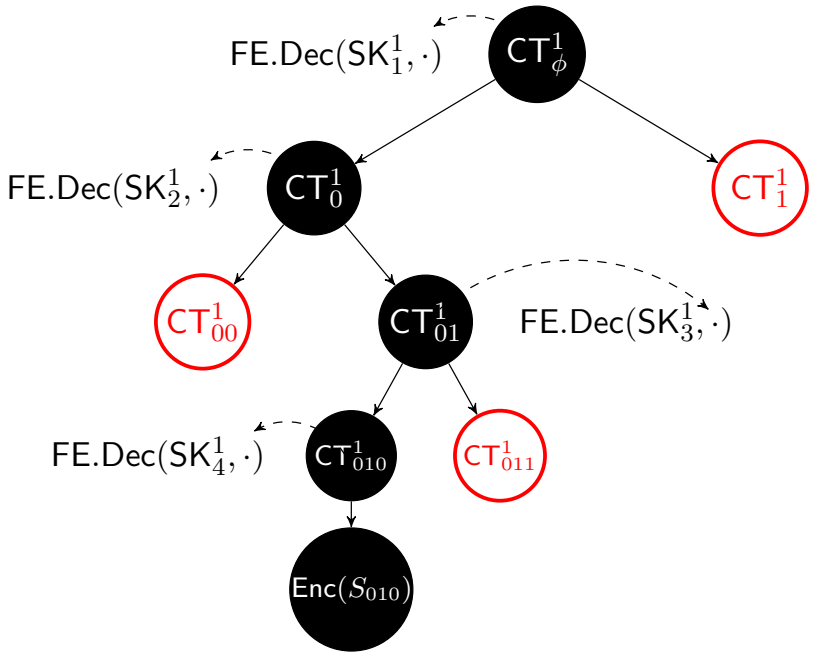

Figure 5: Example of iterated decryption of public key on input 010 and $\kappa = 3$.

Theorem 3.1 *Assuming the security of prefix puncturable PRF, single key selective security of* FE*, the* UFE *construction described in Figure 6 is selectively secure against unbounded collusions.*

Proof Overview. We present an high level overview of the proof.

In the real world the adversary is given an UFE encryption of the message m_b given by $(pk^*, \{\mathsf{L}_{i,(m_b)_i}\}_{i\in[\kappa]})$ where $(pk^*, sk^*) \leftarrow \mathsf{PK.KeyGen}(1^\kappa)$ and $\mathsf{L}_{i,(m_b)_i} := \mathsf{PRF}(S_{pk^*}, i\|(m_b)_i)$. We wish to indistinguishably change to a hybrid that is independent of the challenge bit b. We would using the security of garbling scheme to achieve this purpose. To use the security of garbled circuits we need to do the following:

1. Replace $\{\mathsf{L}_{i,(m_b)_i}\}_{i\in[\kappa]}$ with uniformly chosen random strings in the challenge ciphertext.
2. Replace $\{\mathsf{L}_{i,1-(m_b)_i}\}_{i\in[\kappa]}$ generated in $\mathsf{Output}^2[\Psi^f_{\kappa+1}, f]$ (described in Figure 9) with uniformly chosen random strings for every functional secret key query f.
3. Use uniformly chosen random tape to generate $\{c_{i,b_i}\}_{i\in[\kappa],b_i\in\{0,1\}}$ and $\widetilde{C_f}$ in $\mathsf{Output}^2[\Psi^f_{\kappa+1}, f]$ for every functional secret key query f.
4. Replace $\{c_{i,1-(m_b)_i}\}_{i\in[\kappa]}$ to encrypt some junk value say the all zeroes string.

Once the above steps are accomplished it is easy to observe that the adversary gets access to $\widetilde{C_f}$ and exactly one label (corresponding to bits of m_b) for each input wire for every functional secret key query f. The result now directly follows from the security of garbled circuits. Let us see how the above steps are accomplished.

The first step of our proof strategy is to replace S_{pk^*} with an uniformly chosen random string. We proceed as follows. We start by removing traces of the key S_{pk^*} from the master public key our

- $\mathsf{UFE.KeyGen}(1^\kappa)$:

  1. Sample $S, K_\phi^1 \leftarrow \{0,1\}^\kappa$ to serve as keys for prefix puncturable PRF. PPRF_S works on inputs of length κ. $\mathsf{PPRF}_{K_\phi^1}$ works on inputs of length 2κ.
  2. Sample $sk_1^1, \cdots, sk_{\kappa+1}^1 \leftarrow \mathsf{SK.KeyGen}(1^\kappa)$ (where $|sk_1| = \kappa$) and compute $\Psi_i^1 \leftarrow \mathsf{SK.Enc}(sk_i^1, 0^{\mathsf{len}_i(\kappa)})$ for all $i \in [\kappa+1]$ where $\mathsf{len}_i(\cdot)$ is a length function that would be specified later.
  3. Sample $(\mathsf{PK}_i^1, \mathsf{MSK}_i^1) \leftarrow \mathsf{FE.KeyGen}(1^\kappa)$ for $i \in [\kappa+1]$. Compute $\mathsf{SK}_i^1 \leftarrow \mathsf{FE.KeyGen}(\mathsf{MSK}_i^1, \mathsf{BitExt}_i^1[\Psi_i^1, \mathsf{PK}_{i+1}^1])$ for all $i \in [\kappa]$ and $\mathsf{SK}_{\kappa+1}^1 \leftarrow \mathsf{FE.KeyGen}(\mathsf{MSK}_{\kappa+1}^1, \mathsf{Output}^1[\Psi_{\kappa+1}^1])$ where $\mathsf{BitExt}_i^1[\Psi_i^1, \mathsf{PK}_{i+1}^1]$ is described in Figure 7 and $\mathsf{Output}^1[\Psi_{\kappa+1}^1]$ is described in Figure 8.
  4. Compute $\mathsf{CT}_\phi^1 \leftarrow \mathsf{FE.Enc}(\mathsf{PK}_1^1, (\phi, S, K_\phi^1, 0^{\kappa(\kappa+1)}, 0))$.
  5. Output the master public key PK to be $(\mathsf{CT}_\phi^1, \{\mathsf{SK}_i^1\}_{i\in[\kappa]})$ and the master secret key $\mathsf{MSK} = S$.

- $\mathsf{UFE.Enc}(\mathsf{PK}, m)$:

  1. Sample $(pk, sk) \leftarrow \mathsf{PK.KeyGen}(1^\kappa)$ where $|pk| = \kappa$.
  2. For $i = 1, \cdots, \kappa$ compute: $(\mathsf{CT}^1_{pk_{[i-1]}\|0}, \mathsf{CT}^1_{pk_{[i-1]}\|0}) \leftarrow \mathsf{FE.Dec}(\mathsf{SK}_i^1, \mathsf{CT}^1_{pk_{[i-1]}})$ where $\mathsf{CT}^1_{pk_{[0]}}$ is defined to be CT_ϕ^1.
  3. Compute $c := \mathsf{FE.Dec}(\mathsf{SK}_{\kappa+1}^1, \mathsf{CT}_{pk})$ and recover $S_{pk} = \mathsf{PK.Dec}(sk, c)$.
  4. Compute $\{\mathsf{L}_{i,m_i}\}_{i\in[\kappa]} \leftarrow \mathsf{PRF}(S_{pk}, i\|m)$ where m_i denotes the i-th bit of the message m.
  5. Output $(pk, \{\mathsf{L}_{i,m_i}\}_{i\in[\kappa]})$.

- $\mathsf{UFE.KeyGen}(\mathsf{MSK}, f)$:

  1. Sample $K_\phi^f \leftarrow \{0,1\}^\kappa$ to serve as key for prefix puncturable PRF. $\mathsf{PPRF}_{K_\phi^f}$ works on inputs of length 2κ.
  2. Sample $sk_1^f, \cdots, sk_{\kappa+1}^f \leftarrow \mathsf{SK.KeyGen}(1^\kappa)$ (where $|sk_f| = \kappa$) and compute $\Psi_i^f \leftarrow \mathsf{SK.Enc}(sk_i^f, 0^{\mathsf{len}_i^f(\kappa)})$ for all $i \in [\kappa+1]$ where $\mathsf{len}_i^f(\cdot)$ is a length function that would be specified later.
  3. Sample $(\mathsf{PK}_i^f, \mathsf{MSK}_i^f) \leftarrow \mathsf{FE.KeyGen}(1^\kappa)$ for $i \in [\kappa+1]$.
  4. Compute $\mathsf{SK}_i^f \leftarrow \mathsf{FE.KeyGen}(\mathsf{MSK}_i^f, \mathsf{BitExt}_i^1[\Psi_i^f, \mathsf{PK}_{i+1}^f])$ for all $i \in [\kappa]$ and $\mathsf{SK}_{\kappa+1}^f \leftarrow \mathsf{FE.KeyGen}(\mathsf{MSK}_{\kappa+1}^f, \mathsf{Output}^2[\Psi_{\kappa+1}^f, C_f])$ where $\mathsf{BitExt}_i^1[\Psi_i^f, \mathsf{PK}_{i+1}^f]$ is described in Figure 7, C_f is the description of the circuit computing f and $\mathsf{Output}^2[\Psi_{\kappa+1}^f, C_f]$ is described in Figure 9.
  5. Compute $\mathsf{CT}_\phi^f \leftarrow \mathsf{FE.Enc}(\mathsf{PK}_1^f, (\phi, S, K_\phi^f, 0^{\kappa(\kappa+1)}, 0))$.
  6. Output $\mathsf{SK}_f = (\mathsf{CT}_\phi^f, \{\mathsf{SK}_i^f\}_{i\in[\kappa+1]})$.

Figure 6: Transformation from Single key to Unbounded Key Secure

- $\mathsf{UFE.Dec}(\mathsf{SK}_f, \mathsf{CT})$:
  1. Parse CT as $(pk, \{\mathsf{L}_{i,m_i}\}_{i\in[\kappa]})$
  2. For $i = 1, \cdots, \kappa$ compute
  $$(\mathsf{CT}^f_{(pk)_{[i-1]}\|0}, \mathsf{CT}^f_{(pk)_{[i-1]}\|1}) \leftarrow \mathsf{FE.Dec}(\mathsf{SK}_i, \mathsf{CT}^f_{(pk)_{[i-1]}})$$
  where $\mathsf{CT}^f_{(pk)_{[0]}}$ is defined to be CT^f_{ϕ}.
  3. Compute $\widetilde{C_f}, \{c_{i,b_i}\}_{i\in[\kappa],b_i\in\{0,1\}} \leftarrow \mathsf{FE.Dec}(\mathsf{SK}_{\kappa+1}, \mathsf{CT}^f_{pk})$. Decrypt c_{i,m_i} using L_{i,m_i} as the key and obtain Inp_{i,m_i}.
  4. Output $\mathsf{Garb.Eval}(\widetilde{C_f}, \{\mathsf{Inp}_{i,m_i}\}_{i\in[\kappa]})$.

Figure 6: Transformation from Single key to Unbounded Key Secure

Input. $x \in \{0,1\}^{i-1}$, S_x, K^1_x, $(sk^1_1, \cdots, sk^1_{\kappa+1})$, mode
Constants. Ψ^1_i, PK^1_{i+1}

- If $\mathsf{mode} = 0$
  1. Compute $S_{x\|b} \leftarrow \mathsf{PrefixPunc}(S_x, b)$, $K^1_{x\|b} \leftarrow \mathsf{PrefixPunc}(K^1_x, b\|0)$ and ${K'}^1_{x\|b} \leftarrow \mathsf{PrefixPunc}(K^1_x, b\|1)$ for $b \in \{0,1\}$.
  2. Output $\{\mathsf{FE.Enc}(\mathsf{PK}^1_{i+1}, x\|b, S_{x\|b}, K^1_{x\|b}, sk_1, \mathsf{mode}; {K'}^1_{x\|b})\}_{b\in\{0,1\}}$.
- Else,
  1. Recover $(x\|0, \mathsf{CT}^1_{x\|0})$ and $(x\|1, \mathsf{CT}^1_{x\|1})$ from $\mathsf{SK.Dec}(sk^1_i, \Psi^1_i)$ and output $\{\mathsf{CT}^1_{x\|0}, \mathsf{CT}^1_{x\|1}\}$.

Figure 7: $\mathsf{BitExt}^1_i[\Psi^1_i, \mathsf{PK}^1_{i+1}]$

Input. $x \in \{0,1\}^{\kappa}$, S_x, K^1_x, $(sk^1_1, \cdots, sk^1_{\kappa+1})$, mode
Constants. $\Psi^1_{\kappa+1}$

- If $\mathsf{mode} = 0$, output $\mathsf{PK.Enc}(x, S_x; K^1_x)$.
- Else, recover (x, Val_x) from $\mathsf{SK.Dec}(sk^1_{\kappa+1}, \Psi^1_{\kappa+1})$ and output Val_x.

Figure 8: $\mathsf{Output}^1[\Psi^1_{\kappa+1}]$

Input. x where $x \in \{0,1\}^\kappa$, S_x, K^f_x, $(sk^f_1, \cdots, sk^f_{\kappa+1})$, mode
Constants. $\Psi^f_{\kappa+1}, C_f$

- If $\mathsf{mode} = 0$,
  1. Compute $(\widetilde{C_f}, \{\mathsf{Inp}_{i,b_i}\}_{i\in[\kappa],b_i\in\{0,1\}}) = \mathsf{Garb.Circuit}(C_f; K^f_{x\|0\|0})$.
  2. Compute $\mathsf{L}_{i,b_i} \leftarrow \mathsf{PRF}(S_{pk}, i\|b_i)$ for all $i \in [\kappa]$ and $b_i \in \{0,1\}$.
  3. Compute $c_{i,b_i} \leftarrow \mathsf{SK.Enc}(\mathsf{L}_{i,b_i}, \mathsf{Inp}_{i,b_i}; K^f_{x\|1\|i\|b_i})$ for all $i \in [\kappa]$ and $b_i \in \{0,1\}$. For each bit $i \in [\kappa]$, permute $c_{i,0}$ and $c_{i,1}$ as per the bits of $K^f_{x\|0\|1}$.
  4. Output $\widetilde{C_f}, \{c_{i,b_i}\}_{i\in[\kappa],b_i\in\{0,1\}}$.
- Else, recover (x, Val_x) from $\mathsf{SK.Dec}(sk^f_{\kappa+1}, \Psi^f_{\kappa+1})$ and output Val_x.

Figure 9: $\mathsf{Output}^2[\Psi^f_{\kappa+1}, f]$

UFE scheme. Concretely, we tunnel through the path from the root to the leaf labeled pk^* in the master public key. This ensures that the master public key does not contain S_z where z is a strict prefix of pk^*. But the final value that is output by the master public key on input pk^* is a public key encryption of S_{pk^*} under pk^*. We now rely on the semantic security of public key encryption scheme (under pk^*) to change S_{pk^*} (in the encryption) to some junk value, say the all zeroes string. At this stage the master public key is devoid of any trace of S_{pk^*} or S_z where z is a prefix of pk^*.

Now we tunnel through the path labeled pk^* in each of the function secret keys SK_f to remove traces of S_{pk^*}. Once this is done S_{pk^*} is used only in the generation of the challenge ciphertext and in the computation of the garbled circuit in each of SK_f on input pk^*. We rely on the pseudorandomness at prefix punctured point property of the PRF to replace S_{pk^*} with a random string. This also means that we can replace $\mathsf{L}_{i,b_i} = \mathsf{PRF}(S_{pk^*}, i\|b_i)$ for all $i \in [\kappa], b_i \in \{0,1\}$ used in generating the challenge ciphertext and in generating the encrypted input labels in each of the functional secret key to uniformly chosen random strings. Thus, we have accomplished Steps 1,2.

While tunneling through the root to leaf pk^* path in each of the functional secret keys SK_f we additionally remove traces of $K^f_{pk^*}$ along with S_{pk^*}. This means that we can rely on pseudorandomness at prefix punctured property of PRF to replace the random coins used in generating $\{c_{i,b_i}\}_{i\in[\kappa],b_i\in\{0,1\}}$ and $\widetilde{C_f}$ in every functional secret key with uniformly chosen ones. Thus, Step 3 is accomplished. We now rely on the semantic security of the encryption under $\mathsf{L}_{i,(1-m_b)_i}$ to change encryptions of the actual input label $\mathsf{inp}_{i,(1-(m_b)_i)}$ to encryptions of some junk value say the all zeroes string thus accomplishing Step 4.

Proof of Theorem 3.1 We first set up some notation.

Notation. Let $\mathsf{Prefixes}(x)$ denote the set of all prefixes (κ in number) of the string x. Formally,

$$\mathsf{Prefixes}(x) := \{x_{[i]}\}_{i\in[\kappa]}$$

Let $\mathsf{Siblings}(x)$ denote the set of siblings of all prefixes of x. Formally,

$$\mathsf{Siblings}(x) := \{y_{[i-1]} \| (1-y_i) : \forall y \in \mathsf{Prefixes}(x), i \in [\kappa] \text{ where } |y| = i\}$$

The proof proceeds via a hybrid argument.

- Hyb_0 : In this hybrid the adversary is given the challenge ciphertext encrypting the message m_b. To be more precise, the challenge ciphertext is given by $(pk^*, \{\mathsf{L}_{i,(m_b)_i}\}_{i\in[\kappa]})$ where $(pk^*, sk^*) \leftarrow \mathsf{PK.KeyGen}(1^\kappa)$ and $\mathsf{L}_{i,(m_b)_i} \leftarrow \mathsf{PRF}(S_{pk^*}, i\|(m_b)_i)$ for all $i \in [\kappa]$. All key generation queries are generated as per the construction described in Figure 6.

- Hyb_1 : In this hybrid we are going to "tunnel" through the path from root to the leaf node labeled pk^* in the master public key. This step is realized through a couple of intermediate hybrids.

  Let $P_1 := \mathsf{Prefixes}(pk^*)$ and $Q_1 = \mathsf{Siblings}(pk^*) \triangle P_1$. For every $z \in P_1 \cup Q_1$ let CT^1_z be the result of the iterated decryption procedure on the master public key with z as input. Additionally, let $\mathsf{Val}^1_{pk^*}$ be the output of the decryption of $\mathsf{CT}^1_{pk^*}$ under $\mathsf{SK}^1_{\kappa+1}$. Let

  $$\mathsf{str}_i = \|_{z\in P_1\cup Q_1 \wedge |z|=i}(z, \mathsf{CT}_z)$$

  $$\mathsf{str}_{\kappa+1} = (pk^*, \mathsf{Val}^1_{pk^*})$$

  We set $\mathsf{len}_i(\kappa)$ to be the maximum length of str_i over all choices of pk^*. We pad str_i to this size.

  - $\mathsf{Hyb}_{0,1}$: In this hybrid we are going to change how Ψ^1_i is generated. Instead of encrypting the all zeroes string of length $\mathsf{len}_i(\kappa)$ we encrypt str_i. Indistinguishability follows from the semantic security of the symmetric key encryption since the keys $sk^1_1, \cdots, sk^1_{\kappa+1}$ are not needed to simulate Hyb_0 or $\mathsf{Hyb}_{0,1}$.
  - $\mathsf{Hyb}_{0,2}$: In this hybrid, we change how CT^1_ϕ is generated. Instead of generating CT^1_ϕ to be $\mathsf{FE.Enc}(\mathsf{PK}^1_1, (\phi, S, K^1_\phi, 0^{\kappa(\kappa+1)}, 0))$ we generate it as $\mathsf{FE.Enc}(\mathsf{PK}^1_1, (\phi, 0^\kappa, 0^\kappa, (sk^1_1, \cdots, sk^1_{\kappa+1}), 1))$. We now argue that $\mathsf{Hyb}_{0,2}$ is indistinguishable from $\mathsf{Hyb}_{0,1}$. Notice that output of $\mathsf{BitExt}^1_1[\Psi^1_1, \mathsf{PK}^1_2]$ is same on $(\phi, S, K^1_\phi, 0^{\kappa(\kappa+1)}, 0)$ and $(\phi, 0^\kappa, 0^\kappa, (sk^1_1, \cdots, sk^1_{\kappa+1}), 1)$. Also, the choice of the two messages and the functionality for which the secret key is obtained do not depend on the public parameters. Hence, it follows from the selective security of FE scheme under PK^1_1 that $\mathsf{Hyb}_{0,1}$ and $\mathsf{Hyb}_{0,2}$ are indistinguishable.
  - $\mathsf{Hyb}_{0,3}$: In this hybrid we are going to tunnel through the paths from the root to the leaf labeled pk^*. To achieve this we are going to change CT_z that is encrypted in Ψ^1_1 for every $z \in P_1$. We don't change the encryption when $z \in Q_1$. In particular, we change $\mathsf{CT}^1_z = \mathsf{FE.Enc}(\mathsf{PK}^1_{|z|+1}, (z, S_z, K^1_z, 0^{\kappa(\kappa+1)}, 0); K'^1_z)$ to $\mathsf{FE.Enc}(\mathsf{PK}^1_{|z|+1}, (z, 0^\kappa, 0^\kappa, (sk^1_1, \cdots, sk^1_{\kappa+1}), 1); r_z)$ where r_z is chosen uniformly at random. Notice that as a result S_z for every z that is a strict prefix of pk^* does not appear in the public key of our UFE scheme.

    We introduce an ordering of strings in P_1. For every string $x, y \in P_1$ $x \prec y$ if and only if $|x| \leq |y|$. This induces a partial ordering of the strings in P_1. Notice that $x \prec x$ as a result of this ordering. We let $\mathsf{Hyb}_{0,2,x}$ to denote the hybrid where for all $z \prec x$, CT_z has been changed from $\mathsf{FE.Enc}(\mathsf{PK}^1_{|z|+1}, (z, S_z, K^1_z,$

$0^{\kappa(\kappa+1)}, 0); K'^1_z)$ to $\mathsf{FE.Enc}(\mathsf{PK}^1_{|z|+1}, (z, 0^\kappa, 0^\kappa, (sk^1_1, \cdots, sk^1_{\kappa+1}), 1); r_z)$. We prove for any two adjacent strings x, x' where $x' \prec x$ in ordered P_1 we prove $\mathsf{Hyb}_{0,2,x}$ is indistinguishable to $\mathsf{Hyb}_{0,2,x'}$. Since $|P_1| \leq \kappa$ we get $\mathsf{Hyb}_{0,2}$ is indistinguishable to $\mathsf{Hyb}_{0,2}$ through a series a κ hybrids.

* $\mathsf{Hyb}_{0,2,x',1}$: In this hybrid we change CT^1_x to $\mathsf{FE.Enc}(\mathsf{PK}^1_{|x|+1}, (x, S_x, K^1_x, 0^{\kappa(\kappa+1)}, 0); r_x)$ where r_x is chosen uniformly at random. Notice that for all strings y that are prefixes of x, CT^1_y has already been changed to $\mathsf{FE.Enc}(\mathsf{PK}^1_{|y|+1}, (y, 0^\kappa, 0^\kappa, (sk^1_1, \cdots, sk^1_{\kappa+1}), 1); r_y)$ because $y \prec x$ by our ordering. For every y that is a prefix of x, K^1_y is not needed to simulate $\mathsf{Hyb}_{0,2,x'}$ and $\mathsf{Hyb}_{0,2,x',1}$. It follows from the pseudorandomness at prefix punctured point property of PRF key K^1_ϕ we have $\mathsf{Hyb}_{0,2,x'}$ is indistinguishable to $\mathsf{Hyb}_{0,2,x',1}$. Illustration for this hybrid change is given in Figure 10.

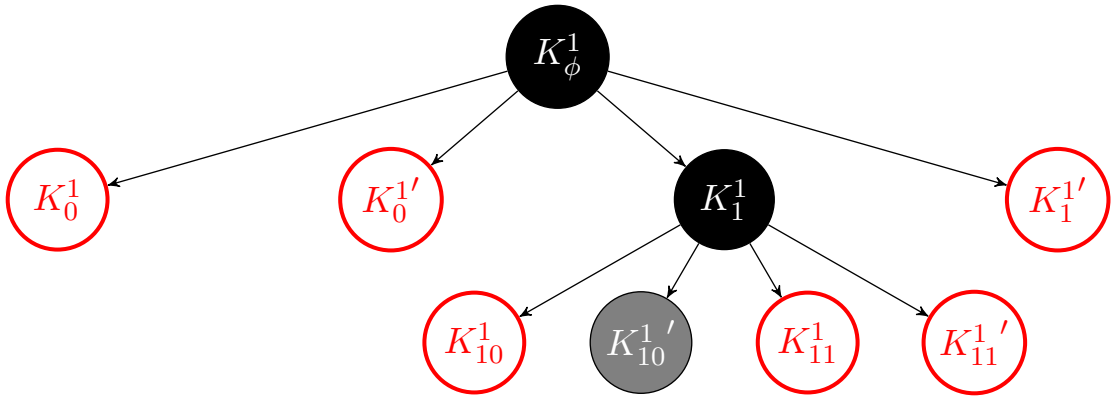

Figure 10: Illustration for $\mathsf{Hyb}_{0,2,x',1}$ where $x' = 1$ and $x = 10$. The blackened nodes are not needed for simulation.

* $\mathsf{Hyb}_{0,2,x',2}$: In this hybrid we change CT_x to $\mathsf{FE.Enc}(\mathsf{PK}^1_{|x|+1}, (x, 0^\kappa, 0^\kappa, (sk^1_1, \cdots, sk^1_{\kappa+1}), 1); r_x)$. Notice that decrypting $\mathsf{FE.Enc}(\mathsf{PK}^1_{|x|+1}, (x, 0^\kappa, 0^\kappa, (sk^1_1, \cdots, sk^1_{\kappa+1}), 1); r_x))$ and $\mathsf{FE.Enc}(\mathsf{PK}^1_{|x|+1}, (x, S_x, K^1_x, 0^{\kappa(\kappa+1)}, 0); r_x)$ under the secret key $\mathsf{SK}^1_{|x|+1}$ has the same output due to the choice of Ψ^*_1. Also, the choice of the two messages and the functionality for which the secret key is obtained do not depend on the public parameters. Hence, it follows from the selective security of FE scheme under $\mathsf{PK}^1_{|x|+1}$ that $\mathsf{Hyb}_{0,2,x',1}$ and $\mathsf{Hyb}_{0,2,x',2}$ are indistinguishable.

Notice that $\mathsf{Hyb}_{0,2,x',2}$ is distributed identically to $\mathsf{Hyb}_{0,2,x}$.

- Hyb_2 : In this hybrid we are going to change Val_{pk^*} encrypted in Ψ^*_1. Notice that in Hyb_2, Val_{pk^*} is set to be an public key encryption of S_{pk^*} under the public key pk^* (using pseudorandomly generated random bits). In this hybrid, we are going to change Val_{pk^*} to be an public key encryption of all zeroes string (0^κ) under pk^*.

  – $\mathsf{Hyb}_{1,1}$: In this hybrid, we generate the randomness used for encrypting S_{pk^*} under the public key pk^* uniformly instead of generating it pseudorandomly using the key $K^1_{pk^*}$. Notice that K^1_z for every z that is a prefix of pk^* is not needed to simulate either Hyb_1

or $\mathsf{Hyb}_{1,1}$. Therefore, from the pseudorandomness at prefix punctured point property of PRF under key K^1_ϕ, Hyb_1 is indistinguishable from $\mathsf{Hyb}_{1,1}$.

- $\mathsf{Hyb}_{1,2}$: In this hybrid, we change Val_{pk^*} to be an encryption of 0^κ under pk^*. Indistinguishability of $\mathsf{Hyb}_{1,1}$ and $\mathsf{Hyb}_{1,2}$ follows from the semantic security of public key encryption.

- Hyb_3 : In this hybrid we are going to tunnel through the paths from the root to the leaf pk^* in each function secret key SK_f that is queried by the adversary. We explain the details for a single function key SK_f and we can extend to all function secret keys by a standard hybrid argument. The indistinguishability argument for a single function secret key SK_f is similar to our argument to show indistinguishability between Hyb_0 and Hyb_1.

  Let $P_2 := \mathsf{Prefixes}(pk^*)$ and $Q_2 = \mathsf{Siblings}(pk^*)$. For every $z \in P_2 \cup Q_2$ let CT^f_z be the result of the iterated decryption procedure on the function secret key SK_f with z as input. Additionally, let $\widetilde{C_f}, \{c_{i,b_i}\}_{i\in[\kappa],b_i\in\{0,1\}}$ be the output of the decryption of CT_{pk^*} under $\mathsf{SK}^f_{\kappa+1}$. Let

$$\mathsf{str}^f_i = \|_{z\in P_2\cup Q_2 \wedge |z|=i}(z, \mathsf{CT}_z)$$

$$\mathsf{str}^f_{\kappa+1} = (pk^*, \widetilde{C_f}, \{c_{i,b_i}\}_{i\in[\kappa],b_i\in\{0,1\}})$$

  We set $\mathsf{len}'_i(\kappa)$ to be the maximum length of str^f_i over all choices of f. We pad str^f_i to this size.

  - $\mathsf{Hyb}_{2,1}$: In this hybrid we are going to change how Ψ^f_i is generated. Instead of encrypting the all zeroes string of length $\mathsf{len}'_i(\kappa)$ we encrypt str^f_i. Indistinguishability follows from the semantic security of the symmetric key encryption since the keys $sk^f_1, \cdots, sk^f_{\kappa+1}$ are not needed to simulate Hyb_2 or $\mathsf{Hyb}_{2,1}$..

  - $\mathsf{Hyb}_{2,2}$: In this hybrid, we change how CT^f_ϕ is generated. Instead of generating CT^f_ϕ to be $\mathsf{FE.Enc}(\mathsf{PK}^f_1, (\phi, S, K^f_\phi, 0^{\kappa(\kappa+1)}, 0))$ we generate it as $\mathsf{FE.Enc}(\mathsf{PK}^f_1, (\phi, 0^\kappa, 0^\kappa, (sk^f_1, \cdots, sk^f_{\kappa+1}), 1))$. We now argue that $\mathsf{Hyb}_{2,2}$ is indistinguishable from $\mathsf{Hyb}_{2,1}$. Notice that output of $\mathsf{BitExt}^f_1[\Psi^*_f, \mathsf{PK}^f_2]$ is same on $(\phi, S, K^f_\phi, 0^{\kappa(\kappa+1)}, 0)$ and $(\phi, 0^\kappa, 0^\kappa, (sk^f_1, \cdots, sk^f_{\kappa+1}), 1)$. Also, the choice of the two messages and the functionality for which the secret key is obtained do not depend on the public parameters. Hence, it follows from the selective security of FE scheme under PK^f_1 that $\mathsf{Hyb}_{2,1}$ and $\mathsf{Hyb}_{2,2}$ are indistinguishable.

  - $\mathsf{Hyb}_{2,3}$: In this hybrid we are going to tunnel through the paths from the root to the leaf labeled pk^* in SK_f. To achieve this we are going to change CT_z that is encrypted in Ψ^f_i for every $z \in P_2$. We don't change the encryption when $z \in Q_2$. In particular, we change $\mathsf{CT}^f_z = \mathsf{FE.Enc}(\mathsf{PK}^f_{|z|+1}, (z, S_z, K^f_z, 0^{\kappa(\kappa+1)}, 0); K'^f_z)$ to $\mathsf{FE.Enc}(\mathsf{PK}^f_{|z|+1}, (z, 0^\kappa, 0^\kappa, (sk^f_1, \cdots, sk^f_{\kappa+1}), 1); r_z)$ where r_z is chosen uniformly at random.

    We introduce an ordering of strings in P_2. For every string $x, y \in P_2$ $x \prec y$ if and only if $|x| \leq |y|$. This induces a partial ordering of the strings in P_2. We let $\mathsf{Hyb}_{2,2,x}$ to denote the hybrid where for all $z \prec x$, CT_z has been changed from $\mathsf{FE.Enc}(\mathsf{PK}^f_{|z|+1}, (z, S_z, K^f_z, 0^{\kappa(\kappa+1)}, 0); K'^f_z)$ to $\mathsf{FE.Enc}(\mathsf{PK}^f_{|z|+1}, (z, 0^\kappa, 0^\kappa, (sk^f_1, \cdots, sk^f_{\kappa+1}), 1); r_z)$. We prove for any

two adjacent strings x, x' where $x' \prec x$ in ordered P_2 we prove $\mathsf{Hyb}_{2,2,x}$ is indistinguishable to $\mathsf{Hyb}_{2,2,x'}$. Since $|P_1| \leq \kappa$ we get $\mathsf{Hyb}_{2,2}$ is indistinguishable to $\mathsf{Hyb}_{2,3}$ through a series a κ hybrids.

* $\mathsf{Hyb}_{2,2,x',1}$: In this hybrid we change CT^f_x to $\mathsf{FE.Enc}(\mathsf{PK}^f_{|x|+1}, (x, S_x, K^f_x, 0^{\kappa(\kappa+1)}, 0); r_x)$ when $|x| \leq \kappa$ where r_x is chosen uniformly at random. Notice that for all strings y that are prefixes of x, CT^f_y has already been changed to $\mathsf{FE.Enc}(\mathsf{PK}^f_{|y|+1}, (y, 0^\kappa, 0^\kappa, (sk^f_1, \cdots, sk^f_{\kappa+1}), 1); r_y)$ because $y \prec x$ by our ordering. For every y that is a prefix of x, K^f_y is not needed to simulate $\mathsf{Hyb}_{2,2,x'}$ and $\mathsf{Hyb}_{2,2,x',1}$. It follows from the pseudorandomness at prefix punctured point property of PRF key K^f_ϕ we have $\mathsf{Hyb}_{2,2,x'}$ is indistinguishable to $\mathsf{Hyb}_{2,2,x',1}$.
* $\mathsf{Hyb}_{2,2,x',2}$: In this hybrid we change CT^f_x to $\mathsf{FE.Enc}(\mathsf{PK}^f_{|x|+1}, (x, 0^\kappa, 0^\kappa, (sk^f_1, \cdots, sk^f_{\kappa+1}), 1); r_x)$. Notice that decrypting $\mathsf{FE.Enc}(\mathsf{PK}^f_{|x|+1}, (x, 0^\kappa, 0^\kappa, (sk^f_1, \cdots, sk^f_{\kappa+1}), 1); r_x))$ and $\mathsf{FE.Enc}(\mathsf{PK}^f_{|x|+1}, (x, S_x, K^f_x, 0^{\kappa(\kappa+1)}, 0); r_x)$ under the secret key $\mathsf{SK}^f_{|x|+1}$ has the same output due to the choice of $\Psi^{|x|+1}_f$. Also, the choice of the two messages and the functionality for which the secret key is obtained do not depend on the public parameters. Hence, it follows from the selective security of FE scheme under $\mathsf{PK}^f_{|x|+1}$ that $\mathsf{Hyb}_{2,2,x',1}$ and $\mathsf{Hyb}_{2,2,x',2}$ are indistinguishable.

Notice that $\mathsf{Hyb}_{2,2,x',2}$ is distributed identically to $\mathsf{Hyb}_{2,2,x}$.

- Hyb_4 : In this hybrid we are going to change S_{pk^*} used to generate the challenge ciphertext to an uniformly chosen random κ-bit string T^*. We observe that for z that is a prefix of pk^*, S_z is not needed to simulate either Hyb_3 or Hyb_4 because we have tunneled through pk^* in the master public key and tunneled through the root to the leaf node pk^* in all the function secret keys SK_f. Hence from the pseudorandomness at prefix punctured point property of the PRF under the key S, Hyb_4 is computationally indistinguishable to Hyb_3. Notice that this also implies from the property of the pseudorandom function that $\{\mathsf{L}_{i,b_i}\}$ for every $i \in [\kappa]$ and for every $b_i \in \{0,1\}$ can be changed to uniformly chosen random strings. This change is made to challenge ciphertext as well as encryption keys used for generating $\{c_{i,b_i}\}_{i\in[\kappa],b_i\in\{0,1\}}$ in $\Psi^f_{\kappa+1}$ in each functional secret key SK_f.
- Hyb_5 : In this hybrid we are going to change to change the randomness used for generating garbled circuit and the encryptions c_{i,b_i} that are encrypted in $\Psi^f_{\kappa+1}$ in each of the function secret keys SK_f to uniformly chosen random strings. Observe that since we have tunneled through pk^* in each of the function secret keys it follows from pseudorandomness of prefix punctured point property of the PRF under the key K^f_ϕ, Hyb_5 is computationally indistinguishable to Hyb_6.
- Hyb_6 : In this hybrid we are going to change $c_{i,1-(m_b)_i}$ to encrypting all zeroes string instead of encrypting $\mathsf{Inp}_{i,1-(m_b)_i}$. This change is made in $\Psi^f_{\kappa+1}$ in each of the function secret keys SK_f. Indistinguishablity of Hyb_5 and Hyb_6 follows from the semantic security of secret key encryption under $\mathsf{L}_{i,1-(m_b)_i}$.
- Hyb_7 : In this hybrid, we are going to change $\{\mathsf{Inp}_{i,(m_b)_i}\}_{i\in[\kappa]}, \widetilde{C_f}$ to be output of the simulator for the garbled circuit. This change is made in $\Psi^f_{\kappa+1}$ in each of the function secret keys SK_f.

More precisely, we set $\{\mathsf{Inp}_{i,(m_b)_i}\}_{i\in[\kappa]}, \widetilde{C_f} \leftarrow \mathsf{Sim}(1^\kappa, C_f, f(m_0))$ (note that $f(m_0) = f(m_b)$). Indistinguishability of Hyb_6 and Hyb_7 follows from the security of garbled circuits.

In Hyb_7 the view of the adversary is independent of the challenge bit b. Hence the advantage that the adversary has in guessing the bit b is 0 in Hyb_7.

4 Efficiency Analysis

In this section we relax the requirement of full compactness from our single-key selectively secure FE scheme to weakly compact ciphertexts. Parts of the efficiency analysis are taken verbatim from Bitansky and Vaikuntanathan [BV15].

Recall that a FE scheme with weakly compact ciphertexts has an encryption circuit whose size grows sub-linearly with the circuit size of functions for which function secret keys are given.

Let $F_1, F_2, \cdots, F_{\kappa+1}$ be the functionalities implemented by the secret keys $\mathsf{SK}_1^f, \cdots, \mathsf{SK}_{\kappa+1}^f$. [6] Notice that for any $i = \{1, \cdots, \kappa\}$, F_i implements the encryption circuit E_{i+1} for the functional encryption scheme under PK_{i+1}, symmetric decryption circuit and a prefix puncturing circuit. The size of the functional encryption circuit and the symmetric decryption circuit is bounded by $|E_{i+1}|.\mathsf{poly}(\kappa)$ and the size of the prefix puncturing circuit is bounded by $\mathsf{poly}(\kappa)$. Therefore,

$$|F_i| \leq |E_{i+1}|.\mathsf{poly}(\kappa)$$

From our assumption that the underlying FE scheme is weakly compact we get:

$$|E_i| \leq |F_i|^{1-\varepsilon}.\mathsf{poly}(\kappa)$$

Notice that:

$$|F_{\kappa+1}| \leq |C_f|.\mathsf{poly}(\kappa)$$

Hence we get:

$$|E_i| \leq |F_i|^{1-\varepsilon}.\mathsf{poly}(\kappa) \leq |E_{i+1}|^{1-\varepsilon}.(\mathsf{poly}(\kappa))^{1-\varepsilon}.\mathsf{poly}(\kappa)$$

By recursively enumerating we get:

$$|E_i| \leq |C_f|^{1-\varepsilon}.\mathsf{poly}(\kappa).\prod_{j=1}^{\kappa+2-i} \mathsf{poly}(\kappa)^{(1-\varepsilon)^j}$$

We observe that:

$$\prod_{j=1}^{\kappa+2-i} \mathsf{poly}(\kappa)^{(1-\varepsilon)^j} \leq \prod_{j=0}^{\infty} \mathsf{poly}(\kappa)^{(1-\varepsilon)^j} \leq (\mathsf{poly}(\kappa))^{\frac{1}{\varepsilon}}$$

Hence, for all $i \in [\kappa+1]$ we get:

$$|E_i| \leq |C_f|^{1-\varepsilon}.\mathsf{poly}(\kappa)^{1+\frac{1}{\varepsilon}}$$

which implies efficiency of our underlying construction.

[6] We restrict our attention to the functional secret keys of our scheme. The analysis of the master public key is exactly the same.

5 Acknowledgements.

The second author would like to thank Divya Gupta, Peihan Miao, Pratyay Mukherjee and Mark Zhandry for insightful discussions.

References

[ABSV15] Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. From selective to adaptive security in functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 657–677, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.

[AFV11] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, pages 21–40, 2011.

[AGVW13] Shweta Agrawal, Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption: New perspectives and lower bounds. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 500–518, 2013.

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 308–326, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.

[AJS15] Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Achieving compactness generically: Indistinguishability obfuscation from non-compact functional encryption. *IACR Cryptology ePrint Archive*, 2015:730, 2015.

[AS16] 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.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, pages 213–229, 2001.

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

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*, pages 501–519, 2014.

[BSW06] Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, pages 573–592, 2006.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography - 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings*, pages 253–273, 2011.

[BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In Venkatesan Guruswami, editor, *56th FOCS*, pages 171–190, Berkeley, CA, USA, October 17–20, 2015. IEEE Computer Society Press.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part II*, pages 280–300, 2013.

[BZ14] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In *Proceedings of CRYPTO 2014*, 2014.

[CFN94] Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*, pages 257–270, 1994.

[CIJ+13] Angelo De Caro, Vincenzo Iovino, Abhishek Jain, Adam O'Neill, Omer Paneth, and Giuseppe Persiano. On the achievability of simulation-based security for functional encryption. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 519–535, 2013.

[Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *Cryptography and Coding, 8th IMA International Conference, Cirencester, UK, December 17-19, 2001, Proceedings*, pages 360–363, 2001.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49, Berkeley, CA, USA, October 26–29, 2013. IEEE Computer Society Press.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[GKP+13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 555–564, 2013.

[GPS15] Sanjam Garg, Omkant Pandey, and Akshayaram Srinivasan. On the exact cryptographic hardness of finding a nash equilibrium. Cryptology ePrint Archive, Report 2015/1078, 2015. http://eprint.iacr.org/2015/1078.

[GPSW06] 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 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006*, pages 89–98, 2006.

[GPSZ16] Sanjam Garg, Omkant Pandey, Akshayaram Srinivasan, and Mark Zhandry. Breaking the sub-exponential barrier in obfustopia. Cryptology ePrint Archive, Report 2016/102, 2016. http://eprint.iacr.org/2016/102.

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 162–179, 2012.

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 545–554, 2013.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 503–523, 2015.

[HJO+15] Brett Hemenway, Zahra Jafargholi, Rafail Ostrovsky, Alessandra Scafuro, and Daniel Wichs. Adaptively secure garbled circuits from one-way functions. *IACR Cryptology ePrint Archive*, 2015:1250, 2015.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 669–684, 2013.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, pages 146–162, 2008.

[LOS+10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings*, pages 62–91, 2010.

[LP09] Yehuda Lindell and Benny Pinkas. A proof of security of yao's protocol for two-party computation. *J. Cryptology*, 22(2):161–188, 2009.

[NWZ16] Ryo Nishimaki, Daniel Wichs, and Mark Zhandry. Anonymous traitor tracing: How to embed arbitrary information in a key. In *Proceedings of EUROCRYPT 2016*, 2016.

[O'N10] Adam O'Neill. Definitional issues in functional encryption. *IACR Cryptology ePrint Archive*, 2010:556, 2010.

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, pages 47–53, 1984.

[SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*, pages 463–472, 2010.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, pages 457–473, 2005.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th ACM STOC*, pages 475–484, New York, NY, USA, May 31 – June 3, 2014. ACM Press.

[Wat15] Brent Waters. A punctured programming approach to adaptively secure functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 678–697, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167, Toronto, Ontario, Canada, October 27–29, 1986. IEEE Computer Society Press.