Indistinguishability Obfuscation from Compact Functional Encryption

Prabhanjan Ananth* Abhishek Jain†

Abstract

The arrival of indistinguishability obfuscation (iO) has transformed the cryptographic landscape by enabling several security goals that were previously beyond our reach. Consequently, one of the pressing goals currently is to construct iO from well-studied standard cryptographic assumptions.

In this work, we make progress in this direction by presenting a reduction from iO to a natural form of public-key functional encryption (FE). Specifically, we construct iO for general functions from any single-key FE scheme for NC^1 that achieves selective, indistinguishability security against sub-exponential time adversaries. Further, the FE scheme should be *compact*, namely, the running time of the encryption algorithm must only be a polynomial in the security parameter and the input message length (and not in the function description size or its output length).

We achieve this result by developing a novel *arity amplification technique* to transform FE for single-ary functions into FE for multi-ary functions (aka multi-input FE). Instantiating our approach with known, non-compact FE schemes, we obtain the first constructions of multi-input FE for constant-ary functions based on standard assumptions.

Finally, as a result of independent interest, we construct a compact FE scheme from randomized encodings for Turing machines and learning with errors assumption.

1 Introduction

The ability to cryptographically obfuscate computer programs holds great prospects for securing the future digital world. While general-purpose program obfuscation remained an elusive goal for several decades, this changed recently with the seminal work of Garg et al. [GGH+13b] who gave the first candidate construction of indistinguishability obfuscation [BGI+01] (iO) for $\mathsf{P/poly}$. Since then, iO has been used to realize several advanced cryptographic primitives that were previously beyond our reach, including deniable encryption [SW14], collusion-resistant functional encryption [GGH+13b], round-optimal multiparty computation [GGHR14], and so on. Indeed, by now, iO has been established as a central hub of cryptography.

The tremendous appeal of iO motivates the goal of constructing it from well-studied, standard cryptographic assumptions. However, not much is known in this direction. The security of candidate iO constructions in the works of [GGH+13b, BGK+14, BR14, AGIS14, Zim15, SZ14, AB15] is proven in the"generic graded encoding model" and lacks a reduction in the standard model. The

*Center for Encrypted Functionalities and Department of Computer Science, UCLA. prabhanjan@cs.ucla.edu

†Department of Computer Science, John Hopkins University. abhishek@cs.jhu.edu. The author is supported in part by NSF CNS-1414023 and in part by a DARPA/ARL Safeware Grant # W911NF-15-C-0213

recent works of Pass et al. [PST14] and Gentry et al. [GLSW14] seek to rectify this situation by constructing $i\mathrm{O}$ from various assumptions on multilinear maps [GGH13a]. In particular, Pass et al. [PST14] reduce the security of their construction to on "uber assumption" on multilinear maps while Gentry et al. [GLSW14] provide a reduction to the "multilinear subgroup elimination assumption" (defined in their paper) on composite-order multilinear maps [CLT13].

Till date, these remain the only known constructions of general-purpose $i\mathrm{O}$. Further, all of them rely on a common cryptographic primitive, namely, multilinear maps. This is an unsatisfactory situation, especially in light of several recent attacks on multilinear maps [CHL+15, GHMS14, BWZ14, CLT14]. This calls for new constructions of $i\mathrm{O}$ from other, more familiar cryptographic primitives.

1.1 This Work

In this work, we make progress in this direction by providing a new construction of $i\mathrm{O}$ based on a natural form of functional encryption (FE). Along the way, we also obtain new results on multi-input FE [GGG+14] that significantly improve upon the prior results.

I. Indistinguishability Obfuscation from Compact FE. Our main result is a reduction from $i\mathrm{O}$ to any public-key functional encryption scheme that satisfies a natural "compactness" requirement on the encryption algorithm. Specifically, we give a construction of $i\mathrm{O}$ for $\mathsf{P/poly}$ from any public-key FE scheme for NC^1 that satisfies the following requirements:

- *Security*: It supports *one* key query and achieves selective, indistinguishability security against sub-exponential time adversaries.
- *Compactness*: For any input message x, the running time of the encryption algorithm is polynomial in the security parameter and the size of x. In particular, it does not depend on the circuit description size or the output length of any function f supported by the scheme.[1] We call such an FE scheme *compact*.

We stress that we do *not* require function hiding property [BRS13, AAB+13] from the underlying FE. Indeed, function-hiding public-key FE already implies $i\mathrm{O}$.

On the use of sub-exponential hardness. Our reliance on sub-exponential hardness of the underlying FE scheme is similar in spirit to the use of sub-exponential hardness assumptions in the work of Gentry et al. [GLSW14]. Indeed, as discussed in their paper, the use of sub-exponential hardness assumptions "seems" inherent to realizing $i\mathrm{O}$. We note, however, that to the best of our knowledge, no formal proof supporting this intuition is known.

On the existence of compact FE. While public-key FE is an extremely well-studied notion, somewhat surprisingly, compact FE has remained largely unexplored. Previously, Goldwasser et al. [GKP+13] studied the notion of "succinct" FE which, informally speaking, requires that the size of any ciphertext must be independent of the function description size. We note, however, that this notion does not preclude dependence on the function output length. Indeed, [GKP+13] focuses on functions with single bit output, and their construction does not achieve our desired compactness property for the case of functions with long output.

[1] The compactness requirement can be further relaxed; see Section 1.3.

Compact FE with simulation-based security is known to be impossible for general functions [AGVW13, CIJ+13]. Concretely, in the case of adaptive simulation security, the impossiblity result holds for a single key and message query. In the selective security case, it holds for a single key and unbounded message queries.[2] However, we stress that for our main result, we only require the underlying compact FE scheme to satisfy *indistinguishability security* in the selective model.

Presently, the only known constructions of compact FE for general functions rely on iO [GGH+13b, Wat14].[3] It contrast, *non-compact* FE can be based on LWE [GKP+13], or even semantically-secure public-key encryption [SS10, GVW12].

We hope that this work will bring attention to the natural goal of compactness in FE and that it will be realized from standard complexity assumptions in the future. With this view, we believe that the results in this work open new doors to the eventual goal of realizing iO from well-studied cryptographic assumptions, possibly avoiding multilinear maps altogether.

II. A Technique for Arity Amplification. At the heart of our results is a novel *technique for arity amplification* in secret-key multi-input functional encryption (MiFE), a notion introduced by Goldwasser et al. [GGG+14]. Specifically, we show how to transform a selectively-secure secret-key MiFE scheme for i-ary functions into another selectively-secure[4] secret-key MiFE scheme for $(i+1)$-ary functions. Interestingly, we achieve this by "knitting together" a secret-key FE scheme for i-ary functions with a public-key FE scheme for 1-ary functions. In order to prove the security of our transformation, we build on program puncturing techniques that were first introduced by Sahai and Waters [SW14] in the context of iO and recently developed in the context of secret-key FE by Brakerski and Segev [BS15] and Komargodski et al. [KSY15].

Starting from a secret key FE scheme for single-ary functions (aka single-input FE) and applying our transformation *iteratively*, we obtain a secret-key MiFE scheme for multi-ary functions. This iterated procedure is sensitive to the efficiency of the underlying single-input FE and yields different end results depending upon whether the underlying FE scheme is compact or not.

More concretely, given a compact *single-key* FE scheme for NC^1, we first convert it into a compact FE scheme for general functions that supports an a priori bounded polynomial number of key queries. This process involves multiple steps, including the key query amplification step of Gorbunov et al [GVW12] and the generic transformation from [GHRW14, ABSV14] for boosting the function family from NC^1 to general functions. As we discuss in Appendix C, this process preserves the compactness of the underlying FE scheme.

Then, instantiating our iterated approach with a sub-exponentially secure *compact* FE scheme that supports (say) q number of key queries, we obtain a secret-key MiFE scheme for polynomial-arity functions that supports q key queries and q message queries. Instantiating this result for the case of $q = 2$ and combining it with the MiFE to iO transformation of Goldwasser et al. [GGG+14], we obtain iO for general functions.

III. MiFE for Functions with Small Arity from Standard Assumptions. We also analyze our transformation for the case when the underlying FE scheme is *non-compact*. Recall that in such

[2] A related notion of reusable garbled circuits with output-size independence was recently studied by Gentry et al. [GHRW14]. They proved an analogous impossibility result for this notion in the case of simulation security.

[3] The compact FE constructions of [GGH+13b, Wat14], in fact, achieve stronger security than what we require. Specifically, they achieve security against unbounded key queries, while we only require security against a single key query.

[4] Unless stated otherwise, we only consider selectively-secure (Mi)FE schemes in the subsequent discussion.

a scheme, the running time of the encryption algorithm may depend upon the function description size [SS10, GVW12] or its output length [GKP+13].

Bounded-message security from standard assumptions. Starting with a non-compact FE scheme that supports an a priori bounded polynomial (say) q number of key queries, we obtain a secret-key MiFE scheme for *constant-ary* functions that supports q message and q key queries. Instantiating the underlying FE scheme with [SS10, GVW12], we obtain the above result based on *semantically secure public-key encryption*.[5] This significantly improves over the state of the art in this area in terms of security assumptions. In particular, prior constructions of such an MiFE scheme either rely upon iO [GGG+14] or lack a security proof in the standard model [BLR+15].

Unbounded-message security from iO. Starting with a non-compact FE scheme that achieves security against unbounded key queries, we obtain a secret-key MiFE scheme for constant-ary functions that supports unbounded message and key queries.

Presently, known constructions of public-key FE with security against unbounded collusions rely upon iO and one-way functions [GGH+13b, Wat14] or specific assumptions on composite-order multilinear maps [GGHZ14]. Then, instantiating the underlying FE scheme in our construction with [GGH+13b], we obtain a secret-key MiFE scheme for functions with constant arity that supports *unbounded* number of message and key queries based on iO and one-way functions. Previously, such an MiFE scheme [GGG+14] was only known based on differing-inputs obfuscation [BGI+01, BCP14, ABG+13].

On the optimality of our results. It is easy to see that a secret-key MiFE scheme for 2-ary functions that supports a single key query and *unbounded* message queries already implies a secret-key single-input FE scheme that supports *unbounded* key and message queries. This observation is already implicit in [GGG+14].

In light of the above, we note that our results on secret-key MiFE with bounded message queries are essentially *optimal*.

IV. Compact FE from Randomized Encodings for Turing Machines. Our final contribution is a construction of a single-key, compact FE scheme from the learning with errors (LWE) assumption and randomized encodings (RE) [IK00, AIK06] for Turing machines where the size of the encoding only depends on the size of the Turing machine (TM) and not on its running time or the output length. Combining this with our reduction from iO to compact FE, we get a construction of iO for general circuits from sub-exponentially secure RE for Turing machines and LWE.

Randomized encodings for circuits are known to exist from only one-way functions [Yao86]. In contrast, the problem of RE for TMs has received far less attention. Recently, a few works [LO13, GHL+14, GLOS15] construct RE for RAM programs from only one-way functions; however, the size of the garbled RAM program in these schemes is proportional to the (worst-case) running time of the underlying RAM program. Even more recently, [BGL+15, CHJV15, KLW15] give constructions of RE for TMs where the encoding size is independent of the running time of TM. However, all of these results are based on iO.

[5] At the cost of further decreasing the efficiency of encryption and restricting our attention to a single key query, we can, in fact, obtain this result based on only *one-way functions*. This requires a slight modification in our construction and proof. In particular, to obtain this result, we must replace the underlying public-key FE with a secret-key FE and then leverage the "one-shot" proof technique discussed in Section 1.2. We defer the details to the full version of the paper.

We hope that our work will bring more attention to this natural goal, and that it can be realized from standard cryptographic assumptions in the future.

1.2 Our Techniques

Main goal: Arity Amplification. The starting point of our $i\mathrm{O}$ construction is the recent work of Goldwasser et al. [GGG+14] who showed a transformation from secret-key MiFE to $i\mathrm{O}$. Concretely, [GGG+14] proved that secret-key MiFE for $(n+1)$-ary functions that supports a *single* key query and 2 message queries implies $i\mathrm{O}$ for all functions with input length n. Very roughly, in order to obfuscate a function f with input length n, their idea is to use the first MiFE ciphertext to hide the function and use the remaining n positions to encode f's input domain *à la* Yao's garbled circuits [Yao86]. This, coupled with a secret key for the universal circuit yields an indistinguishability obfuscation of f.

Given their result, our goal of constructing general-purpose $i\mathrm{O}$ from public-key single-input FE reduces to the task of constructing secret-key MiFE scheme for *polynomial-ary* functions from a public-key FE for *single-ary* functions. To help the presentation, we ignore the succinctness requirements on the underlying FE for now, and revisit it later.

At a first glance, it is not clear at all how to proceed towards realizing the above goal.

Knitting together two FE instances. Towards that end, let us first consider a weaker goal of constructing secret-key MiFE for 2-ary functions. Roughly speaking, our main idea is to "knit" together an instance of a *secret-key* single-input FE scheme with an instance of *public-key* single-input FE to obtain a secret-key MiFE for 2-ary functions. Here, the importance of using *both* a secret-key FE and a public-key FE will become clear once we explain our approach.

More concretely, the 2-ary MiFE scheme is constructed as follows:

- The master secret key of the 2-ary scheme consists of a key pair $(\mathsf{pk}, \mathsf{msk})$ of the underlying public-key FE scheme as well as a master secret key msk' of the underlying secret-key FE scheme. Further, a key for a function f is computed as a key K_f of the underlying public-key FE scheme for f.
- In order to encrypt a message m_1 corresponding to the *first* position, we generate (using msk') a function key of the underlying secret-key FE scheme for the following function $\mathcal{G}^{\mathsf{enc}}_{[m_1,K,pk]}$: it contains the message m_1, a key for a pseudorandom function (PRF) K, and the public key pk hardwired in its description. On input a message (m_2, tag), $\mathcal{G}^{\mathsf{enc}}_{[m_1,K,pk]}$ outputs an encryption (using pk) of the combined message $m_1 \| m_2$ w.r.t. the underlying public-key FE. Here, the randomness r for encryption is derived as $r \leftarrow \mathsf{PRF}_K(\mathsf{tag})$.

  A message m_2 corresponding to the *second* position is encrypted (along with a random tag) using the encryption algorithm of the underlying secret-key FE scheme.
- In order to decrypt a pair of ciphertexts (c_1, c_2) using a function key K_f, we first decrypt c_2 using c_1 (recall that c_1 corresponds to a function key of the secret key FE scheme) to produce a new ciphertext $\tilde{c}$ corresponding to the underlying public-key FE scheme. Finally, we decrypt $\tilde{c}$ using K_f to get the desired output.

The correctness of the above construction is easy to verify. A careful reader, however, may immediately notice a security problem. Note that in order to prove security, we must ensure that the first ciphertext hides the message m_1 and the PRF key K. However, this is not necessarily guaranteed by the above construction.

We solve this problem by building upon the recent elegant result of Brakerski and Segev [BS15] who give a generic transformation from any single-input secret-key FE scheme into another secret-key FE scheme that satisfies *function hiding*. Specifically, instead of using a standard secret-key FE, we will use a function-hiding secret-key FE in the above construction. We then rely upon the function-hiding property of the function key to argue that m_1 and K remain hidden. As we will see later, this technique, when generalized to the MiFE setting, is vital to our overall approach.

We highlight another subtle point in the above construction: suppose that we want the 2-ary MiFE scheme to support $q \geq 2$ message queries. Then, since the function keys of the underlying secret-key FE scheme act as ciphertexts in the 2-ary MiFE scheme, we need the underlying secret-key FE scheme to, in fact, support q key queries. To obtain such an FE scheme, we leverage [GVW12] to transform a single-key FE scheme into a q-key FE scheme. We refer the reader to Appendix C for more details.

Overview of proof strategy. Proving the security of the above construction turns out to be quite non-trivial. Suppose that we wish to prove security for q message queries (for each position), say $\{x_i^0, y_i^0\}_{i=1}^q$, $\{x_i^1, y_i^1\}_{i=1}^q$. Further, for simplicity, let us restrict our attention to a single function key query f. One plausible proof strategy would be construct a sequence of roughly q hybrids where at any step $i \in [q]$, we switch from (x_i^0, y_i^0) to (x_i^1, y_i^1). However, note that in the case of MiFE, an adversary can compute "cross-terms" from the challenge message pairs. That is, the adversary is allowed to compute (x_i^b, y_i^b) for any $i, j \in [q]$. Indeed, this is why the security definition of MiFE requires that $f(x_i^0, y_j) = f(x_i^1, y_j)$ for all $i, j \in [q]$. However, note that in the above proof strategy, the adversary might end up computing $f(x_i^1, y_j^0)$ which will allow him to distinguish between two successive hybrids.

A plausible solution to overcome the above problem is to argue indistinguishability in *one shot*. That is, instead of arguing indistinguishability one message-pair at a time, we instead switch all the challenge message pairs corresponding to challenge bit 0 with the ones corresponding to challenge bit 1. Implementing this strategy successfully, however, will require "hardwiring" and "unhardwiring" of the (public-key) encryption of *all* the q^2 message pairs (x_i^b, y_j^b) (each of which corresponds to a different output) in the challenge ciphertexts for the first position that correspond to function keys of the underlying secret-key FE scheme. While this is tolerable for the case of arity 2 (and more generally for constant arity), it quickly becomes prohibitive for large arity. Indeed, for arity $n = \mathsf{poly}(\lambda)$, the number of possible outputs (and therefore the message pair combinations) is exponential.

We solve the above problems by carefully employing a "one-input-at-a-time" strategy where we consider roughly q^2 intermediate hybrids (and q^n in the case of arity n; see below). Very briefly, our proof involves careful hardwiring and un-hardwiring of the (public-key) encryption of each of the q^2 message pairs (x_i^b, y_j^b), *one at a time*, in the challenge ciphertexts for the first position that correspond to function keys of the underlying secret-key FE scheme. Furthermore, we crucially ensure that the adversary cannot learn an output of the form $f(x_i^0, y_j^1)$ at any point in the hybrids. In order to implement these ideas, we rely upon *program puncturing techniques* that were originally introduced in the context of $i\mathrm{O}$ [SW14] and recently developed in the secret-key FE setting by [BS15, KSY15]. In particular, as in the work of [KSY15], we rely on function hiding property of the underlying secret-key FE scheme to argue indistinguishability of these core hybrids. We finally note that our proof strategy bears resemblance to the proof methodology in several recent works [GLW14, GLSW14, CLTV15, BGL+15, CHJV15, KLW15].

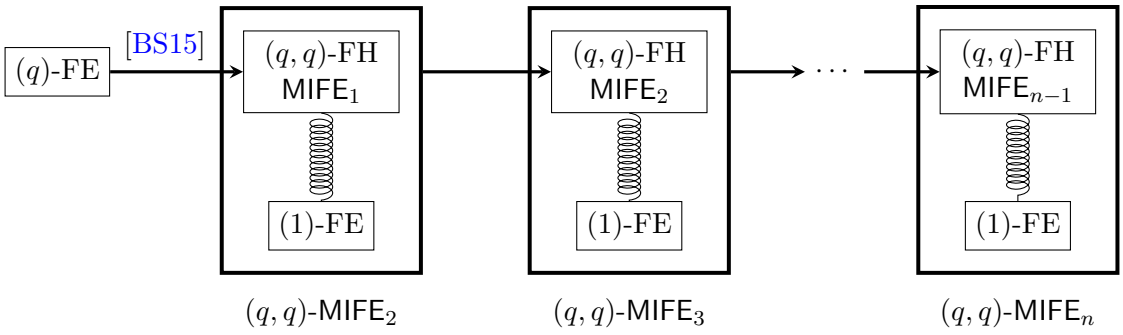

Figure 1 The Iterated Construction. (q)-FE denotes a single-input public-key FE scheme that supports q key queries. (q_1, q_2)-MIFE_i denotes a secret-key MiFE scheme for i-ary functions that supports q_1 key and q_2 message queries. Finally, FH refers to function hiding.

Note that in the above proof strategy, it was crucial that we use a *public-key* FE in our construction. To see this, suppose we were to replace the public-key FE with an instance of a secret-key FE, referred to as $\mathcal{FE}$ (while the other secret-key FE instance used in the construction is referred to as $\mathcal{FE}'$). Note that now, the challenge ciphertexts corresponding to the first position would contain the master secret key (say) msk of $\mathcal{FE}$. Then, in order to execute the aforementioned proof strategy, it would seem that we need to somehow "puncture" msk such that it allows encryption all messages except a select message (say) $x_i^b \| y_j^b$. Furthermore, the punctured msk *should not allow generation of any function keys*, except K_f. However, it is not clear how to realize such a notion of secret-key FE. By using public-key FE, we are able to bypass the above difficulties since by definition, the public key does not need to be hidden.

Climbing the arity ladder. The above approach can be generalized to transform a secret-key MiFE scheme for i-ary functions into a secret-key MiFE scheme for $(i+1)$-ary functions. Concretely, this transformation consists of two steps: first, by using ideas from[BS15], we add function privacy property to the i-ary MiFE scheme. Next, we combine the resultant scheme with a "fresh" instance of a public-key single-input FE scheme to obtain an $(i+1)$-ary MiFE scheme.

In more detail, as in the 2-ary case, the ciphertext corresponding to the first position will consist of a function key of the underlying (function private) c-ary MiFE scheme for the function $\mathcal{G}^{\mathsf{enc}}_{[m_1,K,pk]}$ which is defined similarly to the 2-ary case, except that here it takes as input messages $m_2, \ldots, m_{i+1}$ (along with random tags) and outputs an encryption (using pk) of the combined message $m_1 \| \ldots \| m_{i+1}$ w.r.t. the underlying public-key FE. The ciphertexts corresponding to remaining i positions will correspond to ciphertexts of the underlying c-ary MiFE scheme. The function key for a function f in the $c+1$-ary scheme will correspond to a key K_f for the same function f of the underlying public-key single-input FE scheme.

By applying the above ideas iteratively, we can transform a secret-key single-input FE into a secret-key multi-input FE. Our iterated construction is depicted in Figure 1. The security of the construction follows along the same lines as discussed above.

The role of compactness. Upon "unrolling" our construction of n-ary MiFE scheme, one can observe that it involves n instances of a single-input FE scheme. Specifically, in the n-ary MiFE

scheme, each of the ciphertexts corresponding to the first $n-1$ positions corresponds to a function key of (a different instance of) a single-input FE, while the ciphertext corresponding to the nth position corresponds to a ciphertext of a single-input FE scheme. The function key at position $n-1$ computes an encryption corresponding to the function key at position $n-2$ which in turn computes an encryption corresponding to the function key at position $n-3$, and so on.

With the above view, it is easy to see that the complexity of the above construction becomes prohibitive for $n = \omega(1)$ when it is instantiated with a non-succinct FE scheme. On the other hand, instantiating the construction with a succinct FE scheme allows us to go all the way to $n = \mathsf{poly}(\lambda)$.

We remark that the above discussion is oversimplified. We refer the reader to the technical parts of the paper for more details.

1.3 Concurrent Work

In a concurrent work, Bitansky and Vaikuntanathan [BV15] also construct iO from compact public-key FE. Their construction is, in fact, essentially the same as our "unrolled" iO construction. They also provide a tighter efficiency analysis of their construction and show that a public-key FE where the running time of encryption algorithm has sublinear dependence on the function size, in fact, suffices. (That is, the running time of the encryption algorithm on a message x must be at most $\mathsf{poly}(\lambda, |x|) \cdot |f|^{1-\epsilon}$ for some constant $\epsilon < 1$.) We note that the same analysis goes through for our construction as well, and thus we can also relax the compactness requirement on the underlying public-key FE.

In another concurrent work, Brakerski, Komargodski and Segev [BKS15] give a construction of adaptively-secure secret-key MiFE for constant-ary functions from secret-key FE. We briefly highlight the main differences between their work and ours. Their construction achieves the stronger notion of adaptive security, while we only focus on selectively secure MiFE since it suffices for our main result on iO. Further, their construction uses secret-key FE while we crucially use public-key FE since it allows us to employ the "one-input-at-a-time" proof strategy that is central to obtaining our iO result.

2 Preliminaries

Throughout the paper, we denote the security parameter by λ. We assume that the reader is familiar with basic cryptographic concepts [Gol09].

Given a PPT sampling algorithm A, we use $x \xleftarrow{\$} A$ to denote that x is the output of A when the randomness is sampled from the uniform distribution.

2.1 Indistinguishability Obfuscation

Here we recall the notion of indistinguishability obfuscation (iO) that was defined by Barak et al. [BGI+01].

Definition 1 (Indistinguishability Obfuscator (iO))**.** *A uniform PPT algorithm iO is called an indistinguishability obfuscator for a circuit class $\{\mathcal{C}_\lambda\}$, where $\mathcal{C}_\lambda$ consists of circuits C of the form $C : \{0,1\}^\lambda \to \{0,1\}$, if the following holds:*

- **Completeness:** *For every $\lambda \in \mathbb{N}$, every $C \in \mathcal{C}_\lambda$, every input $x \in \{0,1\}^\lambda$ (i.e., it belongs to the input space of C), we have that*

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

- **Indistinguishability:** *For any PPT distinguisher D, there exists a negligible function $\mathsf{negl}(\cdot)$ such that the following holds: for all sufficiently large $\lambda \in \mathbb{N}$, for all pairs of circuits $C_0, C_1 \in \mathcal{C}_\lambda$ such that $C_0(x) = C_1(x)$ for all inputs x, we have:*

$$\left|\mathsf{Pr}[D(i\mathrm{O}(\lambda, C_0)) = 1] - \mathsf{Pr}[D(i\mathrm{O}(\lambda, C_1)) = 1]\right| \leq \mathsf{negl}(\lambda)$$

2.2 Puncturable Pseudorandom Functions

A pseudorandom function family F consisting of functions of the form $\mathsf{PRF}_K(\cdot)$, that is defined over input space $\{0,1\}^{\eta(\lambda)}$, output space $\{0,1\}^{\chi(\lambda)}$ and key K in the key space $\mathcal{K}$, is said to be a *μ-secure puncturable PRF family* if there exists a PPT algorithm $\mathsf{Puncture}$ that satisfies the following properties:

- **Functionality preserved under puncturing.** $\mathsf{Puncture}$ takes as input a PRF key K, sampled from $\mathcal{K}$, and an input $x \in \{0,1\}^{\eta(\lambda)}$ and outputs K_x such that for all $x' \neq x$, $\mathsf{PRF}_{K_x}(x') = \mathsf{PRF}_K(x')$.
- **Pseudorandom at punctured points.** For every PPT adversary $(\mathcal{A}_1, \mathcal{A}_2)$ such that $\mathcal{A}_1(1^\lambda)$ outputs an input $x \in \{0,1\}^{\eta(\lambda)}$, consider an experiment where $K \xleftarrow{\$} \mathcal{K}$ and $K_x \leftarrow \mathsf{Puncture}(K, x)$. Then for all sufficiently large $\lambda \in \mathbb{N}$,

$$\left|\mathsf{Pr}[\mathcal{A}_2(K_x, x, \mathsf{PRF}_K(x)) = 1] - Pr[\mathcal{A}_2(K_x, x, U_{\chi(\lambda)}) = 1]\right| \leq \mu(\lambda)$$

  where $U_{\chi(\lambda)}$ is a string drawn uniformly at random from $\{0,1\}^{\chi(\lambda)}$.

As observed by [BW13, BGI14, KPTZ13], the GGM construction [GGM86] of PRFs from one-way functions yields puncturable PRFs.

Theorem 1 ([GGM86, BW13, BGI14, KPTZ13])**.** *If μ-secure one-way functions[6] exist, then for all polynomials $\eta(\lambda)$ and $\chi(\lambda)$, there exists a μ-secure puncturable PRF family that maps $\eta(\lambda)$ bits to $\chi(\lambda)$ bits.*

Remark 1. *Note that we do not insist on μ to be a neglible function in the security parameter. When $\mu = \mathsf{negl}(\lambda)$, then we omit it from the notation and simply refer to puncturable PRF families.*

Remark 2. *In this work, we consider pseudorandom function families where the key space is $\{0,1\}^\lambda$. Such a PRF family can be obtained from any PRF family F with arbitrary key space $\mathcal{K}$ as follows: the key sampling algorithm first draws a random key K from $\{0,1\}^\lambda$ and then uses K as randomness to sample a PRF key from the key space $\mathcal{K}$.*[7]

[6] We say that a one-way function family is μ-secure if the probability of inverting a one-way function, that is sampled from the family, is at most $\mu(\lambda)$.

[7] Here, without loss of generality, we assume that the length of the randomness required by the sampling algorithm of F is λ.

2.3 Public-Key Functional Encryption

Syntax. Let $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$ be ensembles where each $\mathcal{X}_\lambda$, $\mathcal{Y}_\lambda$ are sets of size, functions in λ. Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ be an ensemble where each $\mathcal{F}_\lambda$ is a finite collection of functions. Each function $f \in \mathcal{F}_\lambda$ takes as input a string $x \in \mathcal{X}_\lambda$ and outputs $f(x) \in \mathcal{Y}_\lambda$.

A public-key functional encryption (FE) scheme FE for $\mathcal{F}$ consists of four algorithms ($\mathsf{FE.Setup}$, $\mathsf{FE.KeyGen}$, $\mathsf{FE.Enc}$, $\mathsf{FE.Dec}$):

- **Setup.** $\mathsf{FE.Setup}(1^\lambda)$ is a PPT algorithm that takes as input a security parameter λ and outputs a public key, (master) secret key pair $(\mathsf{FE.pk}, \mathsf{FE.msk})$.
- **Key Generation.** $\mathsf{FE.KeyGen}(\mathsf{FE.msk}, f)$ is a PPT algorithm that takes as input a master secret key $\mathsf{FE.msk}$ and a function $f \in \mathcal{F}_\lambda$ and outputs a functional key $\mathsf{FE}.sk_f$.
- **Encryption.** $\mathsf{FE.Enc}(\mathsf{FE.pk}, x)$ is a PPT algorithm that takes as input a public key $\mathsf{FE.pk}$ and a message $x \in \mathcal{X}_\lambda$ and outputs a ciphertext ct.
- **Decryption.** $\mathsf{FE.Dec}(\mathsf{FE}.sk_f, \mathsf{ct})$ is a deterministic algorithm that takes as input a functional key $\mathsf{FE}.sk_f$ and a ciphertext ct and outputs a string $y \in \mathcal{Y}_\lambda$.

Correctness. There exists a negligible function $\mathsf{negl}(\cdot)$ such that for all sufficiently large $\lambda \in \mathbb{N}$, for every message $x \in \mathcal{X}_\lambda$, and for every function $f \in \mathcal{F}_\lambda$,

$$\Pr\Big[f(m) \leftarrow \mathsf{FE.Dec}\big(\mathsf{FE.KeyGen}(\mathsf{FE.msk}, f), \mathsf{FE.Enc}(\mathsf{FE.pk}, m)\big)\Big] \geq 1 - \mathsf{negl}(\lambda)$$

where $(\mathsf{FE.pk}, \mathsf{FE.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$, and the probability is taken over the random coins of all algorithms.

Selective Security. We recall indistinguishability-based selective security for FE. This security notion is modeled as a game between the challenger and the adversary where the adversary can request functional keys and ciphertexts from the challenger. Specifically, the adversary can submit function queries f to the challenger and receive corresponding functional keys. It can also submit a message query of the form (x_0, x_1) and in response, the challenger encrypts message x_b and sends the ciphertext back to the adversary. The adversary wins the game if she can guess b with probability significantly greater than $1/2$ and if $f(x_0) = f(x_1)$ for all function queries f. The only constraint here is that the adversary has to declare the challenge messages at the beginning of the game itself.

Definition 2 (IND-secure FE)**.** *A public-key functional encryption scheme* $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ *for a function family* $\mathcal{F}$ *is said to be* (q_{key}, μ)*-selectively secure if for any PPT adversary* $\mathcal{A}$ *there exists a function* $\mu(\lambda)$ *such that for all sufficiently large* $\lambda \in \mathbb{N}$*, the advantage of* $\mathcal{A}$ *is*

$$\mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A}} = \Big|\Pr[\mathsf{Expt}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda, 0) = 1] - \Pr[\mathsf{Expt}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda, 1) = 1]\Big| \leq \mu(\lambda),$$

where for each $b \in \{0, 1\}$ *and* $\lambda \in \mathbb{N}$*, the experiment* $\mathsf{Expt}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda, b)$ *is defined as follows:*

1. ***Challenge message query****:* $\mathcal{A}$ *submits a message pair* (x_0, x_1) *to* C*.*

2. *The challenger C computes* $(\mathsf{FE.pk}, \mathsf{FE.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$ *and sends* $\mathsf{FE.pk}$ *to the adversary. It then computes* $\mathsf{ct} = \mathsf{FE.Enc}(\mathsf{FE.msk}, x_b)$ *and sends* ct *to* $\mathcal{A}$.

3. ***Function queries**: The following is repeated up to q_{key} times: $\mathcal{A}$ submits a function query $f \in \mathcal{F}_\lambda$ to C. The challenger C computes the function key $\mathsf{FE}.sk_f \leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.msk}, f)$ and sends it to $\mathcal{A}$.*

4. *If there exists a function query f such that $f(x_0) \neq f(x_1)$, then the output of the experiment is $\perp$. Otherwise, the output of the experiment is b', where b' is the output of $\mathcal{A}$.*

Remark 3 (Unbounded IND-secure FE)**.** *One can consider a strengthening of the above definition where the adversary is allowed to make any unbounded polynomial number of function queries. We refer to this as (poly, μ)-selective security.*

Remark 4. *Note that we do not insist on μ to be a negligible function in the security parameter. However, in the case when $\mu = \mathsf{negl}(\lambda)$, then we simply omit it from the notation and refer to q_{key}-selective security of FE.*

Adaptive security. One can consider a stronger notion of security, called (q_{key}, μ)-*adaptive security*, where the adversary can make the challenge message query and the function queries in any arbitrary order, as long as the total number of function queries is q_{key}. Further, when μ is a negligible function in the security parameter, then we omit it from our notation and simply refer to q_{key}-selective security.

Recently, Ananth et al. [ABSV14] gave a general transformation from selectively secure FE to adaptively secure FE.

2.3.1 Compactness

We now define the notion of *compact* FE that will play a central role in our main result on iO. In a compact FE scheme, the running time of the encryption algorithm only depends on the security parameter and the input message length. In particular, it is independent of the complexity of the function family supported by the FE scheme. Note that a direct consequence of this is that the size of the public key must also be independent of the complexity of the function family.

Definition 3 (Compact FE)**.** *Let $p(\cdot)$ be a polynomial. A (q_{key}, μ)-selectively secure public-key FE scheme $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$, defined for an input space $\mathcal{X} = \{\mathcal{X}_\lambda\}$ and function space $\mathcal{F} = \{\mathcal{F}_\lambda\}$ is said to be **compact** if for all $\lambda \in \mathbb{N}$, the size of any public key $\mathsf{FE.pk}$ is $p(\lambda)$, where $(\mathsf{FE.msk}, \mathsf{FE.pk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$, and the running time of the encryption algorithm $\mathsf{FE.Enc}$, on input 1^λ, $\mathsf{FE.pk}$ and a message $x \in \mathcal{X}_\lambda$, is $p(\lambda, q_{\mathsf{key}}, |x|)$.*

Remark 5. *We can define the notion of unbounded compact FE in the same manner as above except that we now allow the number of key queries made by the adversary in the security game to be an arbitrary polynomial.*

We also consider a weaker version of compact FE that we refer to as semi-compact FE. In a semi-compact FE scheme, the run time of the encryption algorithm depends on the output length of the functions. Equivalently, a semi-compact FE scheme is simply a compact FE scheme when we restrict our attention to functions with single-bit outputs.

Definition 4 (Semi-compact FE)**.** *A compact FE scheme for input space $\mathcal{X} = \{\mathcal{X}_\lambda\}$ and function space $\mathcal{F} = \{\mathcal{F}_\lambda\}$, where $\mathcal{F}_\lambda$ consists of functions with single-bit output, is referred to as a* ***semi-compact*** *FE scheme.*

2.4 Secret-Key Multi-Input Functional Encryption

The notion of multi-input functional encryption was proposed by Goldwasser et al. [GGG+14]. Standard FE only allows for computing on a single ciphertext, i.e., it only supports *single-ary* functions. In contrast, multi-input functional encryption (MiFE) allows for (joint) computation over multiple ciphertexts. In other words, it supports *multi-ary* functions.

Analogous to standard FE, one can consider MiFE in two settings, namely, public-key and secret-key setting.[8] In this work, we will restrict our attention to the *secret-key* setting.

Syntax. Let $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$ be ensembles where each $\mathcal{X}_\lambda$, $\mathcal{Y}_\lambda$ are sets of size, functions in λ. Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ be an ensemble where each $\mathcal{F}_\lambda$ is a finite collection of n-ary functions. Each function $f \in \mathcal{F}_\lambda$ takes as input strings $x_1, \ldots, x_n$, where each $x_i \in \mathcal{X}_\lambda$, and outputs $f(x_1, \ldots, x_n) \in \mathcal{Y}_\lambda$.

An MiFE scheme MIFE_n for n-ary functions $\mathcal{F}$ consists of four algorithms ($\mathsf{MIFE}_n.\mathsf{Setup}$, $\mathsf{MIFE}_n.\mathsf{KeyGen}$, $\mathsf{MIFE}_n.\mathsf{Enc}$, $\mathsf{MIFE}_n.\mathsf{Dec}$) described below:

- **Setup.** $\mathsf{MIFE}_n.\mathsf{Setup}(1^\lambda)$ is a PPT algorithm that takes as input a security parameter λ and outputs the master secret key $\mathsf{MIFE}_n.\mathsf{msk}$.
- **Key Generation.** $\mathsf{MIFE}_n.\mathsf{KeyGen}(\mathsf{MIFE}_n.\mathsf{msk}, f)$ is a PPT algorithm that takes as input the master secret key $\mathsf{MIFE}_n.\mathsf{msk}$ and a function $f \in \mathcal{F}_\lambda$. It outputs a functional key $\mathsf{MIFE}_n.sk_f$.
- **Encryption.** $\mathsf{MIFE}_n.\mathsf{Enc}(\mathsf{MIFE}_n.\mathsf{msk}, m, i)$ is a PPT algorithm that takes as input the master secret key $\mathsf{MIFE}_n.\mathsf{msk}$, a message $x \in \mathcal{X}_\lambda$ and an index $i \in [n]$. It outputs a ciphertext $\mathsf{MIFE}_n.\mathsf{ct}$.

  Here index i signals to the encryption algorithm that message x corresponds to the i^{th} input of functions $f \in \mathcal{F}_\lambda$.
- **Decryption.** $\mathsf{MIFE}_n.\mathsf{Dec}(\mathsf{MIFE}_n.sk_f, \mathsf{MIFE}_n.\mathsf{ct})$ is a deterministic algorithm that takes as input a functional key $\mathsf{MIFE}_n.sk_f$ and a ciphertext $\mathsf{MIFE}_n.\mathsf{ct}$. It outputs a value $y \in \mathcal{Y}_\lambda$.

Remark 6. *From now on, we use the phrase "encryption of m in the i^{th} position" to refer to the process of executing* $\mathsf{MIFE}_n.\mathsf{Enc}$ *on the input* $(\mathsf{MIFE}_n.\mathsf{msk}, m, i)$.

Correctness. There exists a negligible function $\mathsf{negl}(\cdot)$ such that for all sufficiently large $\lambda \in \mathbb{N}$, every n-ary function $f \in \mathcal{F}_\lambda$ and input tuple $(x_1, \ldots, x_n) \in \mathcal{X}_\lambda^n$,

$$\Pr\left[\begin{array}{l}\mathsf{MIFE}_n.\mathsf{msk} \leftarrow \mathsf{MIFE}_n.\mathsf{Setup}\left(1^\lambda\right) \;;\; \mathsf{MIFE}_n.sk_f \leftarrow \mathsf{MIFE}_n.\mathsf{KeyGen}\left(\mathsf{MIFE}_n.\mathsf{msk}, f\right) \;; \\ \quad \mathsf{MIFE}_n.\mathsf{Dec}\left(\mathsf{MIFE}_n.sk_f, \{\mathsf{MIFE}_n.\mathsf{Enc}\left(\mathsf{MIFE}_n.\mathsf{msk}, x_i, i\right)\}_{i=1}^n\right) \neq f\left(x_1, \ldots, x_n\right)\end{array}\right] \leq \mathsf{negl}(\lambda)$$

the probability is taken over the random coins of all the algorithms.

[8] Goldwasser et al. [GGG+14] also define a more general notion of MiFE where there is different encryption key for each input position. When the adversary knows all (resp., none of) the encryption keys, then this notion captures the public-key (resp., secret-key) setting.

Selective Security. We recall indistinguishability-based selective security for MiFE. This security notion is modeled as a game between a challenger C and an adversary $\mathcal{A}$ where the adversary can request for functional keys and ciphertexts from C. Specifically, $\mathcal{A}$ can submit n-ary function queries f and respond with the corresponding functional keys. It can also submit message queries of the form $\big((x_{1,0},\ldots,x_{n,0}),\,(x_{1,1},\ldots,x_{n,1})\big)$ and receive encryptions of messages $x_{i,b}$, for $i \in [n]$, and for some bit $b \in \{0,1\}$. The adversary $\mathcal{A}$ wins the game if she can guess b with probability significantly more than one $1/2$ and if $f(x_{1,0},\ldots,x_{n,0}) = f(x_{1,1},\ldots,x_{n,1})$ for all function queries f and message queries $\big((x_{1,0},\ldots,x_{n,0}),\,(x_{1,1},\ldots,x_{n,1})\big)$. As before, the constraint here is that the adversary has to declare the messages at the beginning of the game itself.

Definition 5 (IND-secure MiFE)**.** *A secret-key MiFE scheme* MIFE_n *for n-ary functions* $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in[\mathbb{N}]}$ *and message space* $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in[\mathbb{N}]}$ *is* $(q_{\mathsf{key}}, q_{\mathsf{msg}}, \mu)$*-selectively secure if for any PPT adversary* $\mathcal{A}$*, there exists a function* $\mu(\lambda)$ *such that for all sufficiently large* $\lambda \in \mathbb{N}$*, the advantage of* $\mathcal{A}$ *is*

$$\mathsf{Adv}^{\mathsf{MIFE}_n}_{\mathcal{A}} = \Big|\mathsf{Pr}[\mathsf{Expt}^{\mathsf{MIFE}_n}_{\mathcal{A}}(1^\lambda,0)=1] - \mathsf{Pr}[\mathsf{Expt}^{\mathsf{MIFE}_n}_{\mathcal{A}}(1^\lambda,1)=1]\Big| \le \mu(\lambda),$$

where for each $b \in \{0,1\}$ *and* $\lambda \in \mathbb{N}$*, the experiment* $\mathsf{Expt}^{\mathsf{MIFE}_n}_{\mathcal{A}}(1^\lambda, b)$ *is defined below:*

1. ***Challenge message queries**: $\mathcal{A}$ submits q_{msg} queries, $\Big\{\big((x^j_{1,0},x^j_{1,1}),\ldots,(x^j_{n,0},x^j_{n,1})\big)\Big\}_{j\in[q_{\mathsf{msg}}]}$, with $x^j_{i,0} \in \mathcal{X}_\lambda$, to the challenger C.*

2. *C computes $\mathsf{MIFE}_n.\mathsf{msk} \leftarrow \mathsf{MIFE}_n.\mathsf{Setup}(1^\lambda)$. It then computes $\mathsf{MIFE}_n.\mathsf{ct}^j_i \leftarrow \mathsf{MIFE}_n.\mathsf{Enc}(\mathsf{MIFE}_n.\mathsf{msk}, x^j_{i,b})$ for all $i \in [n]$ for all $j \in [q_{\mathsf{msg}}]$. The challenger C then sends $\big\{(\mathsf{MIFE}_n.\mathsf{ct}^j_1,\ldots,\mathsf{MIFE}_n.\mathsf{ct}^j_n)\big\}_{j\in[q_{\mathsf{msg}}]}$ to the adversary $\mathcal{A}$.*

3. ***Function queries**: The following is repeated up to q_{key} times: $\mathcal{A}$ submits a function query $f \in \mathcal{F}_\lambda$ to C. The challenger C computes $\mathsf{MIFE}_n.sk_f \leftarrow \mathsf{MIFE}_n.\mathsf{KeyGen}(\mathsf{MIFE}_n.\mathsf{msk}, f)$ and sends it to $\mathcal{A}$.*

4. *If there exists a function query f and a challenge message query $\big((x_{1,0},\ldots,x_{n,0}),\,(x_{1,1},\ldots,x_{n,1})\big)$ such that $f(x_{1,0},\ldots,x_{n,0}) \neq f(x_{1,1},\ldots,x_{n,1})$, then the output of the experiment is set to $\perp$. Otherwise, the output of the experiment is set to b', where b' is the output of $\mathcal{A}$.*

Remark 7 (Unbounded IND-secure MiFE)**.** *One can also consider a stronger security notion, namely, $(\mathsf{poly},\mathsf{poly},\mu)$-selective security, where the adversary can make any unbounded polynomial number of function and challenge message queries.*

Remark 8. *Note that we do not insist on μ to be a negligible function in the security parameter. However, in the case when $\mu = \mathsf{negl}(\lambda)$, then we simply omit it from the notation and refer to $(q_{\mathsf{key}}, q_{\mathsf{msg}})$-selective security of MiFE.*

Adaptive security. One can consider a stronger notion of security, called *adaptive security*, where the adversary can interleave the challenge message and the function queries in any arbitrary order. Analogous to Definition 5, we can define $(q_{\mathsf{key}}, q_{\mathsf{msg}}, \mu)$-adaptive secure MiFE. Further, when μ is a negligible function in the security parameter, then we omit it from the notation and refer to $(q_{\mathsf{key}}, q_{\mathsf{msg}})$-adaptive secure MiFE.

3 Function Privacy in MiFE

The notion of function privacy in secret-key FE for single-ary functions was recently formalized by Brakerski-Segev [BS15]. They also give a generic transformation from any non function-private FE scheme to a function-private one. We observe that their ideas can be generalized to the MiFE setting.

Below, we first present the definition of function private secret-key MiFE and then give a generic transformation from any (non-function private) MiFE for n-ary functions into a function-private MiFE for n-ary functions.

Definition 6 (Selective Function Private MiFE)**.** *A secret-key MiFE scheme* MIFE_n *for* n*-ary functions* $\mathcal{F}$ *is* $(q_{\mathsf{key}}, q_{\mathsf{msg}}, \mu)$*-selective function private if for any PPT adversary* $\mathcal{A}$*, there exists a function* $\mu(\lambda)$ *such that for all sufficiently large* $\lambda \in \mathbb{N}$*, the advantage of* $\mathcal{A}$ *is*

$$\mathsf{Adv}^{\mathsf{MIFE}_n}_{\mathcal{A}} = \left|\mathsf{Pr}[\mathsf{Expt}^{\mathsf{MIFE}_n}_{\mathcal{A}}(1^\lambda, 0) = 1] - \mathsf{Pr}[\mathsf{Expt}^{\mathsf{MIFE}_n}_{\mathcal{A}}(1^\lambda, 1) = 1]\right| \leq \mu(\lambda),$$

where for each $b \in \{0,1\}$ *and* $\lambda \in \mathbb{N}$*, the experiment* $\mathsf{Expt}^{\mathsf{MIFE}_n}_{\mathcal{A}}(1^\lambda, b)$ *is defined below:*

1. ***Message queries****:* $\mathcal{A}$ *submits* q_{msg} *queries,* $\left\{\left((x^j_{1,0}, x^j_{1,1}), \ldots, (x^j_{n,0}, x^j_{n,1})\right)\right\}_{j\in[q_{\mathsf{msg}}]}$*, with* $x^j_{i,0} \in \mathcal{X}_\lambda$*, to the challenger* C*.*

2. C *computes* $\mathsf{MIFE}_n.\mathsf{msk} \leftarrow \mathsf{MIFE}_n.\mathsf{Setup}(1^\lambda)$*. It then computes* $\mathsf{MIFE}_n.\mathsf{ct}^j_i \leftarrow \mathsf{MIFE}_n.\mathsf{Enc}(\mathsf{MIFE}_n.\mathsf{msk}, x^j_{i,b})$ *for all* $i \in [n]$ *for all* $j \in [q_{\mathsf{msg}}]$*. The challenger* C *then sends* $\left\{(\mathsf{MIFE}_n.\mathsf{ct}^j_1, \ldots, \mathsf{MIFE}_n.\mathsf{ct}^j_n)\right\}_{j\in[q_{\mathsf{msg}}]}$ *to the adversary* $\mathcal{A}$*.*

3. ***Function queries****: The following is repeated up to* q_{key} *times:* $\mathcal{A}$ *submits a function query* $(f_0, f_1) \in \mathcal{F}^2_\lambda$ *to* C*. The challenger* C *computes* $\mathsf{MIFE}_n.sk_f \leftarrow \mathsf{MIFE}_n.\mathsf{KeyGen}(\mathsf{MIFE}_n.\mathsf{msk}, f_b)$ *and sends it to* $\mathcal{A}$*.*

4. *If there exists a function query* (f_0, f_1) *and a challenge message query* $\left((x_{1,0}, \ldots, x_{n,0}), (x_{1,1}, \ldots, x_{n,1})\right)$ *such that* $f_0(x_{1,0}, \ldots, x_{n,0}) \neq f_1(x_{1,1}, \ldots, x_{n,1})$*, then the output of the experiment is set to* $\perp$*. Otherwise, the output of the experiment is set to* b'*, where* b' *is the output of* $\mathcal{A}$*.*

Remark 9. *When* μ *is a negligible function in the security parameter, then we omit it from the notation and simply refer to* $(q_{\mathsf{key}}, q_{\mathsf{msg}})$*-function privacy of MiFE.*

Adaptive Function Privacy. One can consider a stronger notion of security, called *adaptive function privacy*, where the adversary can interleave the function and the message queries. Analogous to Definition 6, we can define $(q_{\mathsf{key}}, q_{\mathsf{msg}}, \mu)$-adaptive function private MiFE. Further, when μ is a negligible function in the security parameter, then we omit it from the notation and refer to $(q_{\mathsf{key}}, q_{\mathsf{msg}})$-adaptive function private MiFE.

3.1 Constructing Function Private MiFE

We now show how to generically transform a (non function-private) MiFE scheme for c-ary functions into a *function-private* MiFE scheme for c-ary functions. The transformation is a direct adaptation

of the elegant function-privacy transformation of Brakerski and Segev [BS15] in the single input setting.

We stress that this transformation preserves the function arity. This is in contrast to [GGG+14] who give a transformation from an MiFE scheme for $(n+1)$ ary functions into a function-private MiFE scheme for n-ary functions.

Notation. Let $\mathsf{NFP} = (\mathsf{NFP.Setup}, \mathsf{NFP.KeyGen}, \mathsf{NFP.Enc}, \mathsf{NFP.Dec})$ be any non function-private MiFE scheme for all c-ary functions. We denote the associated function space to $\mathcal{F}^c$ and the message space to be $\mathcal{X}^c$. Let $\mathsf{Sym} = (\mathsf{Sym.Setup}, \mathsf{Sym.Enc}, \mathsf{Sym.Dec})$ be a standard symmetric encryption scheme. We construct a function-private MiFE scheme $\mathsf{FP} = (\mathsf{FP.Setup}, \mathsf{FP.KeyGen}, \mathsf{FP.Enc}, \mathsf{FP.Dec})$ for all c-ary functions, where the function space is $\mathcal{F}^{\mathsf{fp},c}$ and the message space is $\mathcal{X}^{\mathsf{fp},c}$.

Setup $\mathsf{FP.Setup}(1^\lambda)$**:** On input a security parameter λ, compute $\mathsf{NFP.msk} \leftarrow \mathsf{NFP.Setup}(1^\lambda)$ and sample two symmetric keys $\mathsf{Sym}.K \leftarrow \mathsf{Sym.Setup}(1^\lambda)$ and $\mathsf{Sym}.K' \leftarrow \mathsf{Sym.Setup}(1^\lambda)$. Output $\mathsf{FP.msk} = (\mathsf{NFP.msk}, \mathsf{Sym}.K, \mathsf{Sym}.K')$.

Key Generation $\mathsf{FP.KeyGen}(\mathsf{FP.msk}, f)$**:** On input the master secret key $\mathsf{FP.msk}$ and function $f \in \mathcal{F}_\lambda^{\mathsf{fp},c}$, first parse $\mathsf{FP.msk} = (\mathsf{NFP.msk}, \mathsf{Sym}.K, \mathsf{Sym}.K')$. Next, compute $\mathsf{Sym.ct} \leftarrow \mathsf{Sym.Enc}(\mathsf{Sym}.K, f)$ and $\mathsf{Sym.ct}' \leftarrow \mathsf{Sym.Enc}(\mathsf{Sym}.K', f)$. Finally, compute $\mathsf{NFP}.sk_U \leftarrow \mathsf{NFP.KeyGen}(\mathsf{NFP.msk}, U_{[\mathsf{Sym.ct},\mathsf{Sym.ct}']})$, where $U_{[\mathsf{Sym.ct},\mathsf{Sym.ct}']} \in \mathcal{F}^c$ is defined in Figure 2. Output $\mathsf{FP}.sk_f = \mathsf{NFP}.sk_U$.

$$U_{[\mathsf{Sym.ct},\mathsf{Sym.ct}']}\Big((x_1, x_1', \mathsf{Sym}.K, \mathsf{Sym}.K'), x_2, \ldots, x_c\Big)$$

1. If $\mathsf{Sym}.K \neq \perp$, compute $f \leftarrow \mathsf{Sym.Dec}(\mathsf{Sym}.K, \mathsf{Sym.ct})$. Output $f(x_1, x_2, \ldots, x_c)$.
2. Else, if $\mathsf{Sym}.K' \neq \perp$, compute $f' \leftarrow \mathsf{Sym.Dec}(\mathsf{Sym}.K', \mathsf{Sym.ct}')$ and output $f'(x_1', x_2, \ldots, x_c)$.
3. Else, output $\perp$.

Figure 2

Encryption $\mathsf{FP.Enc}(\mathsf{FP.msk}, x, i)$**:** On input master secret key $\mathsf{FP.msk}$ and a message $x \in \mathcal{X}_\lambda^{\mathsf{fp},c}$, parse $\mathsf{FP.msk} = (\mathsf{NFP.msk}, \mathsf{Sym}.K, \mathsf{Sym}.K')$.

1. If $i = 1$ then compute $\mathsf{NFP.ct} \leftarrow \mathsf{NFP.Enc}(\mathsf{NFP.msk}, (x, \perp, \mathsf{Sym}.K, \perp), 1)$. Output $\mathsf{NFP.ct}$.
2. Else if $i \neq 1$ then compute $\mathsf{NFP.ct} \leftarrow \mathsf{NFP.Enc}(\mathsf{NFP.msk}, x, i)$. Output $\mathsf{NFP.ct}$.

Decryption $\mathsf{FP.Dec}(\mathsf{FP}.sk_f, \mathsf{FP.ct}_1, \ldots, \mathsf{FP.ct}_c)$**:** On input a functional key $\mathsf{FP}.sk_f = \mathsf{NFP}.sk_U$, and ciphertexts $\mathsf{FP.ct}_i = \mathsf{NFP.ct}_i$ for $i \in [c]$, compute $y \leftarrow \mathsf{NFP.Dec}(\mathsf{NFP}.sk_U, \mathsf{NFP.ct}_1, \ldots, \mathsf{NFP.ct}_c)$. Output y.

This completes the description of the scheme.

Correctness. We argue the correctness of the above scheme. From the correctness of the non function-private MiFE scheme NFP, we have that $\mathsf{NFP.Dec}(\mathsf{NFP}.sk_U, \mathsf{NFP.ct}_1, \ldots, \mathsf{NFP.ct}_c)$ yields the output $y = U_{[\mathsf{Sym.ct},\mathsf{Sym.ct}']}((x_1, \perp, \mathsf{Sym}.K, \perp), x_2, \ldots, x_c)$, where $\mathsf{NFP}.sk_U \leftarrow \mathsf{NFP.KeyGen}$ $(\mathsf{NFP.msk},$

$U_{[\mathsf{Sym.ct},\mathsf{Sym.ct}']}$), $\mathsf{NFP.ct}_1 \leftarrow \mathsf{NFP.Enc}(\mathsf{NFP.msk}, (x_1, \perp, \mathsf{Sym}.K, \perp), 1)$ and $\mathsf{NFP.ct}_i \leftarrow \mathsf{NFP.Enc}(\mathsf{NFP.msk}, x_i, i)$ for $i \in \{2, \ldots, c\}$. From the description of U, we have that $y = f(x_1, \ldots, x_c)$, as desired.

Theorem 2. *Assuming $(q_{\mathsf{key}}, q_{\mathsf{msg}}, \epsilon)$-selective (resp., adaptive) security of* NFP*, pseudorandom function family* F*, and symmetric encryption scheme* Sym*, the proposed scheme* FP *is a $(q_{\mathsf{key}}, q_{\mathsf{msg}}, \frac{\epsilon}{4})$-selective (resp., adaptive) function-private MiFE scheme for all c-ary functions.*

The proof of the above theorem follows along the lines of [BS15]. We give a proof sketch in Appendix A.

4 Our Transformation: From c-ary to (c + 1)-ary MiFE

In this section, we show how to transform a secret-key MiFE scheme for c-ary functions into an MiFE scheme for $(c+1)$-ary functions, for $c \geq 1$.

Our transformation proceeds in two steps:

1. Starting with an MiFE scheme for c-ary functions, we first apply the function privacy transformation from Section 3.1 to obtain a function private MiFE scheme MIFE_c for c-ary functions.

2. Next, we convert MIFE_c into an MiFE scheme MIFE_{c+1} for $c+1$-ary functions. We refer to this step as the *arity amplification* step.

We now describe the arity amplification step. We construct an MiFE scheme for $c+1$-ary functions MIFE_{c+1} with function space $\mathcal{F}^{c+1}$ and message space $\mathcal{X}^{c+1}$.

Notation. We use the following tools in our transformation: (a) A function private MIFE scheme for c-ary functions, denoted as $\mathsf{MIFE}_c = (\mathsf{MIFE}_c.\mathsf{Setup}, \mathsf{MIFE}_c.\mathsf{KeyGen}, \mathsf{MIFE}_c.\mathsf{Enc}, \mathsf{MIFE}_c.\mathsf{Dec})$. Let $\mathcal{F}^{\mathsf{fp},c}$ and $\mathcal{X}^{\mathsf{fp},c}$ be the associated function space and message space, respectively. (b) A public-key FE scheme for single-ary functions, denoted as $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$. Let $\mathcal{F}^{\mathsf{fe}}$ and $\mathcal{X}^{\mathsf{fe}}$ be the associated function space and message space, respectively. (c) A puncturable pseudorandom function family, denoted as $\mathsf{F} = \mathsf{PRF}_K(\cdot)$.

Setup $\mathsf{MIFE}_{c+1}.\mathsf{Setup}(1^\lambda)$: On input a security parameter λ, sample a master secret key $\mathsf{MIFE}_c.\mathsf{msk} \leftarrow \mathsf{MIFE}_c.\mathsf{Setup}(1^\lambda)$ of MIFE_c and a key pair $(\mathsf{FE.pk}, \mathsf{FE.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$ of FE. Output $\mathsf{MIFE}_{c+1}.\mathsf{msk} = (\mathsf{MIFE}_c.\mathsf{msk}, \mathsf{FE.pk}, \mathsf{FE.msk})$.

Key Generation $\mathsf{MIFE}_{c+1}.\mathsf{KeyGen}(\mathsf{MIFE}_{c+1}.\mathsf{msk}, f)$: On input master secret key $\mathsf{MIFE}_{c+1}.\mathsf{msk}$ and a function $f \in \mathcal{F}^{c+1}$, parse $\mathsf{MIFE}_{c+1}.\mathsf{msk} = (\mathsf{MIFE}_c.\mathsf{msk}, \mathsf{FE.pk}, \mathsf{FE.msk})$. Sample a functional key $\mathsf{FE}.sk_f \leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.msk}, f)$ of FE for function f. Output $\mathsf{MIFE}_{c+1}.sk_f = \mathsf{FE}.sk_f$.

Encryption $\mathsf{MIFE}_{c+1}.\mathsf{Enc}(\mathsf{MIFE}_{c+1}.\mathsf{msk}, x, i)$: On input master secret key $\mathsf{MIFE}_{c+1}.\mathsf{msk}$, message $x \in \mathcal{X}^{c+1}$ and index i, parse $\mathsf{MIFE}_{c+1}.\mathsf{msk} = (\mathsf{MIFE}_c.\mathsf{msk}, \mathsf{FE.pk}, \mathsf{FE.msk})$.

1. If $i = 1$, then draw a PRF key $K \in \{0,1\}^\lambda$ at random. Initialize the index vector $\mathcal{I} = (0, \ldots, 0)$. Compute $\mathsf{MIFE}_c.sk_G \leftarrow \mathsf{MIFE}_c.\mathsf{KeyGen}(\mathsf{MIFE}_c.\mathsf{msk}, G)$ where the circuit $G = \mathsf{GenCT}^{(c)}_{[x,1,K,\mathsf{FE.pk},\mathcal{I}]} \in \mathcal{F}^{\mathsf{fp},c}$ is described in Figure 3. Output the ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct} = \mathsf{MIFE}_c.sk_G$.

2. Else, if $2 \leq i \leq c+1$, then perform the following steps:

   - If the input message x is of the form $(x_1, x_2, 1, \tau, i-1)$ then compute $\mathsf{MIFE}_{c+1}.\mathsf{ct} \leftarrow \mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x_1, x_2, 1, \tau, i), i)$
   - Else, choose a tag $\tau \in \{0,1\}^\lambda$ at random. Compute $\mathsf{MIFE}_{c+1}.\mathsf{ct} \leftarrow \mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x, x, 1, \tau, i), i)$.

   Output the ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}$.

$\underline{\mathsf{GenCT}^{(c)}_{[x_1, j_1, K, \mathsf{FE.pk}, \mathcal{I}]}}$

Input: $\Big((x_2, x_2', j_2, \tau_2, i_2), \ldots, (x_{c+1}, x_{c+1}', j_{c+1}, \tau_{c+1}, i_{c+1})\Big)$.

- $\forall k \in \{2, \ldots, c+1\}$, if $i_k \neq k$, then output $\perp$.
- $\mathsf{FE.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.pk}, (x_1, \ldots, x_{c+1}); \mathsf{PRF}(K, \tau_2 || \cdots || \tau_{c+1}))$.
- Output $\mathsf{FE.ct}$.

Figure 3

Decryption $\mathsf{MIFE}_{c+1}.\mathsf{Dec}(\mathsf{MIFE}_{c+1}.sk_f, \mathsf{MIFE}_{c+1}.\mathsf{ct}_1, \ldots, \mathsf{MIFE}_{c+1}.\mathsf{ct}_{c+1})$: On input $(\mathsf{MIFE}_{c+1}.sk_f, \mathsf{MIFE}_{c+1}.\mathsf{ct}_1, \ldots, \mathsf{MIFE}_{c+1}.\mathsf{ct}_{c+1})$, perform the following steps:

1. Parse: (a) $\mathsf{MIFE}_{c+1}.sk_f = \mathsf{FE}.sk_f$, (b) $\mathsf{MIFE}_{c+1}.\mathsf{ct}_1 = \mathsf{MIFE}_c.sk_G$, and (c) $\mathsf{MIFE}_{c+1}.\mathsf{ct}_i = \mathsf{MIFE}_c.\mathsf{ct}_{i-1}$ for all $i \neq 1$, where $\mathsf{MIFE}_c.\mathsf{ct}_{i-1}$ denotes the ciphertext corresponding to $(i-1)^{th}$ position in MIFE_c.
2. Next, compute $\mathsf{FE.ct}^* \leftarrow \mathsf{MIFE}_c.\mathsf{Dec}(\mathsf{MIFE}_c.sk_G, \mathsf{MIFE}_c.\mathsf{ct}_1, \ldots, \mathsf{MIFE}_c.\mathsf{ct}_c)$.
3. Finally, compute $y \leftarrow \mathsf{FE.Dec}(\mathsf{FE}.sk_f, \mathsf{FE.ct}^*)$. Output y.

This completes the description of the scheme.

Remark 10. *The circuits $\mathsf{GenCT}^{(c)}$, $\mathsf{HybGenCT}^{(c,1)}$ (described later in Figure 4), $\mathsf{HybGenCT}^{(c,2)}$ (Figure 5), $\mathsf{HybGenCT}^{(c,3)}$ (Figure 6), $\mathsf{HybGenCT}^{(c,4)}$ (Figure 7) are all suitably padded so that their sizes are the same.*

Correctness. We now argue the correctness of MIFE_{c+1}. Let $\mathsf{MIFE}_c.sk$ be a valid functional key for the function $\mathsf{GenCT}[x_1, j_1, K, \mathsf{FE.pk}, \mathcal{I}]$ w.r.t. MIFE_c. For $i \in [c]$, let $\mathsf{MIFE}_c.\mathsf{ct}_i$ be a valid encryption of x_{i+1} w.r.t. MIFE_c. By the correctness of $\mathsf{MIFE}_c.\mathsf{Enc}$, we have that the output of $\mathsf{MIFE}_c.\mathsf{Dec}(\mathsf{MIFE}_c.sk_{\mathsf{GenCT}}, \mathsf{MIFE}_c.\mathsf{ct}_1, \ldots, \mathsf{MIFE}_c.\mathsf{ct}_c)$ is $\mathsf{FE.ct}^*$, where $\mathsf{FE.ct}^*$ is a valid encryption of $(x_1, \ldots, x_{c+1})$ w.r.t. FE. Further, from the correctness of FE, it follows that the output of $\mathsf{FE.Dec}(\mathsf{FE}.sk_f, \mathsf{FE.ct}^*)$ is $f(x_1, \ldots, x_{c+1})$, where $\mathsf{FE}.sk_f$ is a valid functional key of f w.r.t. FE.

5 Security of $c+1$-ary MiFE

We prove the security of the $c+1$-ary MiFE construction presented in Section 4.

Theorem 3. *The proposed scheme* MIFE_{c+1} *is* (q, q, δ)*-selective secure assuming* $\left(q, \frac{\delta}{8+5q^c}\right)$*-selective security of* FE*,* $\left(\frac{\delta}{8+5q^c}\right)$*-selective security of* F *and* $\left(\frac{\delta}{8+5q^c}\right)$*-selective security of the function-private scheme* MIFE_c*.*

Proof. We prove the theorem by considering a sequence of hybrids. The first hybrid Hybrid_0 corresponds to the real experiment where the challenger chooses the challenge bit b to be 0. In the final hybrid Hybrid_4, the challenger chooses the challenge bit b to be 1. We then argue the indistinguishability of the intermediate hybrids using the primitives in the theorem statement which proves that Hybrid_4 is computationally indistinguishable from Hybrid_1, which proves the theorem.

We define the advantage of an adversary $\mathcal{A}$ (with single-bit output) in the i^{th} hybrid, denoted by $\mathsf{Adv}^i_{\mathcal{A}}$, to be the probability that $\mathcal{A}$ outputs 1 in the i^{th} hybrid.

We introduce some notation that will be make the hybrids easier to describe. Consider the group $(\mathbb{Z}_q^m, +)$, where $q, m \in \mathbb{N}\backslash\{0\}$. The operation '+' associated to this group, maps two vectors $(\mathbf{u}, \mathbf{v}) \in \left(\mathbb{Z}_q^m\right)^2$ and outputs $\mathbf{w}$, where $\mathbf{w}$ is the sum of $\mathbf{u}$ and $\mathbf{v}$ when they are represented as numbers in base-q notation. We define the predicate $\mathsf{LessThan}$ as follows. It takes an input of the form $(\mathcal{I}, \mathcal{I}', m, q)$, where $\mathcal{I} = (j_1, \ldots, j_m)$, $\mathcal{I}' = (j'_1, \ldots, j'_m) \in \mathbb{Z}_q^m$ and outputs 1 if there exists ζ with $1 \leq \zeta \leq m$ such that $j_\zeta < j'_\zeta$[9] and for all $1 \leq k < \zeta$, $j_k = j'_k$ and outputs 0, otherwise.

We now proceed to describe the hybrids. We highlight the main changes in every hybrid by underlining them in red .

Hybrid_1: This corresponds to the real experiment of MIFE_{c+1} where the challenger uses the challenge bit $b = 0$.

1. Adversary submits q number of message queries to the challenger. Denote $(x^j_{i,0}, x^j_{i,1})$ to be the j^{th} message query corresponding to the i^{th} position, where $j \in [q], i \in [c+1]$.
2. Challenger executes $\mathsf{MIFE}_c.\mathsf{Setup}(1^\lambda)$ to obtain $\mathsf{MIFE}_c.\mathsf{msk}$. It then executes $\mathsf{FE.Setup}(1^\lambda)$ to obtain $(\mathsf{FE.pk}, \mathsf{FE.msk})$. Finally, the challenger sets $\mathsf{MIFE}_{c+1}.\mathsf{msk}$ to be $(\mathsf{MIFE}_c.\mathsf{msk}, \mathsf{FE.pk}, \mathsf{FE.msk})$.
3. The challenger then generates the challenge ciphertexts as follows. For every $i \in [c+1], j \in [q]$,
   (a) If $i = 1$:
   - It draws the PRF key $K \in \{0,1\}^\lambda$ at random.
   - It initializes $\mathcal{I}$ to be $(0, \ldots, 0)$.
   - It then executes $\mathsf{MIFE}_c.\mathsf{KeyGen}(\mathsf{MIFE}_c.\mathsf{msk}, G)$ to obtain $\mathsf{MIFE}_c.sk_G$, where $G = \mathsf{GenCT}^{(c)}_{[x^j_{1,0},1,K,\mathsf{FE.pk},\mathcal{I}]}$ is defined in Figure 3.
   - It sends the challenge ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^* = \mathsf{MIFE}_c.sk_G$ to the adversary.

   (b) Else if $2 \leq i \leq c+1$:
   - It picks τ at random.

[9] We emphasize that j_ζ has to be *strictly* less than j'_ζ.

- If $x^j_{i,0}$ is of the form $(x_1, x_2, 1, \tau, i-1)$ then it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x_1, x_2, 1, \tau, i), i)$ to obtain $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$. Otherwise, it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x^j_{i,0}, x^j_{i,0}, 1, \tau, i), i)$ to obtain $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$.
- The ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$ is then sent to the adversary.

4. For every function query f, the challenger first executes $\mathsf{FE.KeyGen}(\mathsf{FE.msk}, f)$ to obtain $\mathsf{FE}.sk_f$. It then sends the functional key $\mathsf{MIFE}_{c+1}.sk_f$, which is set to be $\mathsf{FE}.sk_f$, to the adversary.

5. Adversary outputs b'.

$\underline{\mathsf{Hybrid}_2}$: For every position i, the challenge ciphertexts will now also carry along with them their corresponding query number. That is, for every position i, the encrypted message in the challenge ciphertext corresponding to the j^{th} message query contains j in it. In addition, the challenge ciphertext corresponding to a message query (x_0, x_1) contains both x_0 and x_1 in it instead of just x_0 as in Hybrid_0. Finally, the function $\mathsf{HybGenCT}^{(c,1)}$ is used instead of $\mathsf{GenCT}^{(c)}$ while encrypting messages corresponding to the first position.

1. Adversary submits q number of message queries to the challenger. Denote $(x^j_{i,0}, x^j_{i,1})$ to be the j^{th} message query corresponding to the i^{th} position, where $j \in [q], i \in [c+1]$.

2. Challenger executes $\mathsf{MIFE}_c.\mathsf{Setup}(1^\lambda)$ to obtain $\mathsf{MIFE}_c.\mathsf{msk}$. It then executes $\mathsf{FE.Setup}(1^\lambda)$ to obtain $(\mathsf{FE.pk}, \mathsf{FE.msk})$. Finally, the challenger sets $\mathsf{MIFE}_{c+1}.\mathsf{msk}$ to be $(\mathsf{MIFE}_c.\mathsf{msk}, \mathsf{FE.pk}, \mathsf{FE.msk})$.

3. The challenger then generates the challenge ciphertexts as follows. For every $i \in [c+1], j \in [q]$,

   (a) If $i = 1$ then,
   - It draws the PRF key $K \in \{0,1\}^\lambda$ at random.
   - $\underline{\text{It initializes } \mathcal{I} \text{ to be } (0, \ldots, 0) \in \mathbb{Z}_q^{c+1}}$, where q is the number of message queries made by the adversary.
   - It then executes $\mathsf{MIFE}_c.\mathsf{KeyGen}(\mathsf{MIFE}_c.\mathsf{msk}, G)$ to obtain $\mathsf{MIFE}_c.sk_G$, where $G = \underline{\mathsf{HybGenCT}^{(c,1)}_{[x^j_{1,0}, x^j_{1,1}, j, K, \mathsf{FE.pk}, \mathcal{I}]}}$ is defined in Figure 4.
   - It sends the challenge ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^* = \mathsf{MIFE}_c.sk_{\mathsf{HybGenCT}^{(c,1)}}$ to the adversary.

   (b) Else if $2 \le i \le c+1$ then,
   - It picks τ at random.
   - If $x^j_{i,0}$ is of the form $(x_1, x_2, j, \tau, i-1)$ then it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x_1, x_2, j, \tau, i), i)$ to obtain $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$. Otherwise, it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, \underline{(x^j_{i,0}, x^j_{i,1}, j, \tau, i), i)}$ to obtain the ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$, which is sent to the adversary.

4. For every function query f, the challenger first executes $\mathsf{FE.KeyGen}(\mathsf{FE.msk}, f)$ to obtain $\mathsf{FE}.sk_f$. It then sends the functional key $\mathsf{MIFE}_{c+1}.sk_f = \mathsf{FE}.sk_f$ to the adversary.

5. Adversary outputs b'.

$\underline{\mathsf{HybGenCT}^{(c,1)}_{[x_1,x'_1,j_1,K,\mathsf{FE.pk},\mathcal{I}]}}$

Input: $\Big((x_2,x'_2,j_2,\tau_2,i_2),\ldots,(x_{c+1},x'_{c+1},j_{c+1},\tau_{c+1},i_{c+1})\Big)$.

- $\forall k \in \{2,\ldots,c+1\}$, if $i_k \neq k$, then output $\perp$
- $\mathcal{I}' = (j_1-1,\ldots,j_{c+1}-1)$
- If $\mathsf{LessThan}(\mathcal{I}',\mathcal{I},c+1,q)=1$ then // $\mathcal{I}'$ *is strictly less than* $\mathcal{I}$
  - $\mathsf{FE.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.pk},(x'_1,\ldots,x'_{c+1});\mathsf{PRF}(K,\tau_2||\cdots||\tau_{c+1}))$
- Else,
  - $\mathsf{FE.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.pk},(x_1,\ldots,x_{c+1});\mathsf{PRF}(K,\tau_2||\cdots||\tau_{c+1}))$
- Output $\mathsf{FE.ct}$.

Figure 4

Lemma 1. *For any PPT adversary $\mathcal{A}$, we have $|\mathsf{Adv}^1_{\mathcal{A}}-\mathsf{Adv}^2_{\mathcal{A}}| \leq \epsilon$ assuming the ϵ-selective function privacy of MIFE_c.*

Proof. We design a reduction $\mathcal{B}$ that internally executes $\mathcal{A}$ and breaks the security of MIFE_c with advantage $\delta = |\mathsf{Adv}^1_{\mathcal{A}} - \mathsf{Adv}^2_{\mathcal{A}}|$. The reduction interacts with $\mathcal{A}$ by simulating the role of the challenger of MIFE_{c+1}.

The reduction $\mathcal{B}$ executes $\mathsf{FE.Setup}(1^\lambda)$ to obtain $(\mathsf{FE.pk}^*,\mathsf{FE.msk}^*)$. It then receives message queries $(x^j_{i,0},x^j_{i,1})$, for $i \in [c+1], j \in [q]$, from the adversary $\mathcal{A}$. The reduction $\mathcal{B}$ answers the message query of the form $(x^j_{i,0},x^j_{i,1})$ as follows. We note that $\mathcal{B}$ first generates ciphertexts corresponding to the position $i \neq 1$ and only after that it generates the ciphertexts corresponding to the position $i = 1$. The reason is that $\mathcal{B}$ makes the message queries to the challenger first and only then it makes the functional queries.

1. If $i \neq 1$ then: The reduction first picks $\tau_{i,j}$ at random. It then prepares the messages $x^{i,j}_0 = (x^j_{i,0},x^j_{i,0},1,\tau_{i,j},i)$ and $x^{i,j}_1 = (x^j_{i,0},x^j_{i,1},j,\tau_{i,j},i)$. It then sends all the message pairs $(x^{i,j}_0,x^{i,j}_1)$, for every $i \neq 1$, $j \in [q]$, to the challenger of MIFE_c. Upon receiving the ciphertexts $\mathsf{MIFE}_c.\mathsf{ct}^*_{i,j}$, it sets $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*_{i,j} = \mathsf{MIFE}_c.\mathsf{ct}^*_{i,j}$. The reduction then sends the ciphertexts $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*_{i,j}$, for $i \neq 1, j \in [q]$ to the adversary.

2. Else if $i = 1$ then: The reduction first draws PRF key K at random. It then initializes $\mathcal{I} = (0,\ldots,0)$. It then initializes $\mathcal{I}' = (0,\ldots,0) \in \mathbb{Z}_q^{c+1}$, where the length of the elements (by suitable padding) in each component is λ. It then constructs the circuits $C_j = \mathsf{GenCT}^{(c)}_{[x^j_{1,0},1,K,\mathsf{FE.pk}^*,\mathcal{I}]}$ and $C'_j = \mathsf{HybGenCT}^{(c,1)}_{[x^j_{1,0},x^j_{1,1},j,K,\mathsf{FE.pk}^*,\mathcal{I}']}$. It then submits the function query (C_j,C'_j), for every $j \in [q]$, to the challenger of the security game of MIFE_c. Upon receiving the functional key $\mathsf{MIFE}_c.sk$, it sets $\mathsf{MIFE}_{c+1}.\mathsf{ct}^* = \mathsf{MIFE}_c.sk$ and sends $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$ to the adversary.

For every function query f submitted by the adversary, the reduction $\mathcal{B}$ executes $\mathsf{FE.KeyGen}($

$\mathsf{FE.msk}^*, f)$ to obtain $\mathsf{FE}.sk_f$, which it sends to the adversary $\mathcal{A}$. Finally, the reduction outputs b', where b' is the output of $\mathcal{A}$.

We first claim that $\mathcal{B}$ is a valid adversary in the function privacy game of MIFE_c. Firstly, $\mathcal{B}$ declares all message queries at the beginning of the game itself. Secondly, for every $j_1, \ldots, j_{c+1} \in [q]$, we have

$$C_0\big((x^{j_2}_{2,0}, x^{j_2}_{2,0}, 1, \tau_{2,j_2}, 2), \ldots, (x^{j_{c+1}}_{c+1,0}, x^{j_{c+1}}_{c+1,0}, 1, \tau_{c+1,j_{c+1}}, c+1)\big)$$
$$= C_1\big((x^{j_2}_{2,0}, x^{j_2}_{2,1}, j_2, \tau_{2,j_2}, 2), \ldots, (x^{j_{c+1}}_{c+1,0}, x^{j_{c+1}}_{c+1,1}, j_{c+1}, \tau_{c+1,j_{c+1}}, c+1)\big),$$

where $C_0 = \mathsf{GenCT}^{(c)}_{[x^{j_1}_{1,0}, 1, K, \mathsf{FE.pk}^*, \mathcal{I}]}$ and $C_1 = \mathsf{HybGenCT}^{(c,1)}_{[x^{j_1}_{1,0}, x^{j_1}_{1,1}, j_1, K, \mathsf{FE.pk}^*, \mathcal{I}']}$. In both the cases, encryption of $(x^{j_1}_{1,0}, \ldots, x^{j_{c+1}}_{c+1,0})$ with respect to $\mathsf{FE.pk}^*$ and randomness $\mathsf{PRF}(K, \tau_2^{j_2} || \cdots || \tau_{c+1}^{j_{c+1}})$ is produced. This shows that $\mathcal{B}$ is a valid adversary.

Suppose the challenger of MIFE_c uses the challenge bit 0 to generate the functional keys and the challenge ciphertexts then we are in Hybrid_0. By this, we mean that the challenger upon receiving a function query (resp., message query) of the form (f_0, f_1) (resp., (x_0, x_1)), generates the functional key of f_0 (resp., x_0). Otherwise if the challenger uses the challenge bit 1 then we are in Hybrid_1. This proves our claim. □

$\underline{\mathsf{Hybrid}_3}$: The challenge ciphertext corresponding to each message query (x_0, x_1) is now an encryption of x_1 and it does not contain any information about x_0. The ciphertexts are computed according to Hybrid_2.

1. Adversary submits q number of message queries to the challenger. Denote $(x^j_{i,0}, x^j_{i,1})$ to be the j^{th} message query corresponding to the i^{th} position, where $j \in [q], i \in [c+1]$.
2. Challenger executes $\mathsf{MIFE}_c.\mathsf{Setup}(1^\lambda)$ to obtain $\mathsf{MIFE}_c.\mathsf{msk}$. It then executes $\mathsf{FE.Setup}(1^\lambda)$ to obtain $(\mathsf{FE.pk}, \mathsf{FE.msk})$. Finally, the challenger sets $\mathsf{MIFE}_{c+1}.\mathsf{msk}$ to be $(\mathsf{MIFE}_c.\mathsf{msk}, \mathsf{FE.pk}, \mathsf{FE.msk})$.
3. The challenger then generates the challenge ciphertexts as follows. For every $i \in [c+1], j \in [q]$,
   (a) If $i = 1$:
   - It draws the PRF key $K \in \{0,1\}^\lambda$ at random.
   - It initializes $\mathcal{I}$ to be $(0, \ldots, 0) \in \mathbb{Z}_q^{c+1}$, where q is the number of message queries made by the adversary.
   - It then executes $\mathsf{MIFE}_c.\mathsf{KeyGen}(\mathsf{MIFE}_c.\mathsf{msk}, G)$ to obtain $\mathsf{MIFE}_c.sk_G$, where $G = \underline{\mathsf{HybGenCT}^{(c,1)}_{[x^j_{1,1}, x^j_{1,1}, j, K, \mathsf{FE.pk}, \mathcal{I}]}}$ is defined in Figure 4.
   - It sends the challenge ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^* = \mathsf{MIFE}_c.sk_{\mathsf{HybGenCT}^{(c,1)}}$ to the adversary.

   (b) Else if $2 \le i \le c+1$:
   - It picks τ at random.
   - If $x^j_{i,1}$ is of the form $(x_1, x_2, j, \tau, i-1)$ then it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x_1, x_2, j, \tau, i), i)$ to obtain $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$. Otherwise, it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, \underline{(x^j_{i,1}, x^j_{i,1}, j, \tau, i)}, i)$ to obtain the ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$, which is sent to the adversary.

4. For every function query f, the challenger first executes $\mathsf{FE.KeyGen}(\mathsf{FE.msk}, f)$ to obtain $\mathsf{FE}.sk_f$. It then sends the functional key $\mathsf{MIFE}_{c+1}.sk_f = \mathsf{FE}.sk_f$ to the adversary.

5. Adversary outputs b'.

Lemma 2. *For any PPT adversary $\mathcal{A}$, we have $|\mathsf{Adv}^2_{\mathcal{A}} - \mathsf{Adv}^3_{\mathcal{A}}| \leq (6+5q^c)\epsilon$ assuming the ϵ-selective function privacy of MIFE_c, ϵ-selective security of FE and ϵ-security of F.*

This lemma forms the crux of our overall proof. We prove it later in Section 5.1.

$\underline{\mathsf{Hybrid}_4}$: This corresponds to the real experiment where the challenger uses the challenge bit 1.

1. Adversary submits q number of message queries to the challenger. Denote $(x^j_{i,0}, x^j_{i,1})$ to be the j^{th} message query corresponding to the i^{th} position, where $j \in [q], i \in [c+1]$.

2. Challenger executes $\mathsf{MIFE}_c.\mathsf{Setup}(1^\lambda)$ to obtain $\mathsf{MIFE}_c.\mathsf{msk}$. It then executes $\mathsf{FE.Setup}(1^\lambda)$ to obtain $(\mathsf{FE.pk}, \mathsf{FE.msk})$. Finally, the challenger sets $\mathsf{MIFE}_{c+1}.\mathsf{msk}$ to be $(\mathsf{MIFE}_c.\mathsf{msk}, \mathsf{FE.pk}, \mathsf{FE.msk})$.

3. The challenger then generates the challenge ciphertexts as follows. For every $i \in [c+1], j \in [q]$,

   (a) $i = 1$:
   - It draws the PRF key $K \in \{0,1\}^\lambda$ at random.
   - $\underline{\text{It initializes } \mathcal{I} \text{ to be } (0, \ldots, 0).}$
   - It then executes $\mathsf{MIFE}_c.\mathsf{KeyGen}(\mathsf{MIFE}_c.\mathsf{msk}, G)$ to obtain $\mathsf{MIFE}_c.sk_G$, where $G = \underline{\mathsf{GenCT}^{(c)}_{[x^j_{1,1},1,K,\mathsf{FE.pk},\mathcal{I}]}}$ is defined in Figure 3.
   - It sends the challenge ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^* = \mathsf{MIFE}_c.sk_{\mathsf{HybGenCT}}$ to the adversary.

   (b) $2 \leq i \leq c+1$:
   - It picks τ at random.
   - If $x^j_{i,1}$ is of the form $\underline{(x_1, x_2, 1, \tau, i-1)}$ then it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x_1, x_2, 1, \tau, i), i)$ to obtain $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$. Otherwise, it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, \underline{(x^j_{i,1}, x^j_{i,1}, 1, \tau, i), i)}$ to obtain the ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$, which is sent to the adversary.

4. For every function query f, the challenger first executes $\mathsf{FE.KeyGen}(\mathsf{FE.msk}, f)$ to obtain $\mathsf{FE}.sk_f$. It then sends the functional key $\mathsf{MIFE}_{c+1}.sk_f = \mathsf{FE}.sk_f$ to the adversary.

5. Adversary outputs b'.

Lemma 3. *For any PPT adversary $\mathcal{A}$, we have $|\mathsf{Adv}^3_{\mathcal{A}} - \mathsf{Adv}^4_{\mathcal{A}}| \leq \epsilon$ assuming the ϵ-selective function privacy game of MIFE_c.*

The proof of the above lemma is similar to the proof of the Lemma 1. We omit the details.
Assuming ϵ-security of FE, ϵ-security of puncturable pseudorandom functions, ϵ-security of MIFE_c, in Lemmas 1,2, and 3 we have that for any PPT adversary $\mathcal{A}$,

$$|\mathsf{Adv}^1_{\mathcal{A}} - \mathsf{Adv}^4_{\mathcal{A}}| \leq (8 + 5q^{c+1})\epsilon$$

By setting $\epsilon = \frac{\delta}{8+5q^{c+1}}$, we have the proof of the theorem. □

5.1 Proof of Lemma 2

We first give a sequence of intermediate hybrids between Hybrid_2 and Hybrid_3. Then we establish computational indistinguishability of every two consecutive intermediate hybrids which establishes the indistinguishability of Hybrid_2 and Hybrid_3. In more detail, we define the hybrids $\mathsf{Hybrid}_{2,\vec{j}.i}$, for all $\vec{j} \in [q]^{c+1}$, $i \in [5]$. We also define a sequence of hybrids $\mathsf{Hybrid}_{2^+.i}$, for $i \in [5]$. We then prove that the following equations are true, assuming the existence of ϵ-selective function privacy of MIFE_c, ϵ-security of FE and ϵ-security of F.

1. $|\mathsf{Adv}^2_{\mathcal{A}} - \mathsf{Adv}^{2.(1,\ldots,1).1}_{\mathcal{A}}| \leq \epsilon$.
2. $|\mathsf{Adv}^{2.\vec{j}.i}_{\mathcal{A}} - \mathsf{Adv}^{2.\vec{j}.i+1}_{\mathcal{A}}| \leq \epsilon$, for all $\vec{j} \in [q]^{c+1}$ and $i \in [4]$.
3. $|\mathsf{Adv}^{2.\vec{j}.5}_{\mathcal{A}} - \mathsf{Adv}^{2.\vec{j}+1.1}_{\mathcal{A}}| \leq \epsilon$, for all $\vec{j} \in [q]^{c+1}$ and $\vec{j} \neq (q,\ldots,q)$.
4. $|\mathsf{Adv}^{2.(q,\ldots,q).5}_{\mathcal{A}} - \mathsf{Adv}^{2^+.1}_{\mathcal{A}}| \leq \epsilon$.
5. $|\mathsf{Adv}^{2^+.i}_{\mathcal{A}} - \mathsf{Adv}^{2^+.i+1}_{\mathcal{A}}| \leq \epsilon$, for all $i \in [4]$.
6. $|\mathsf{Adv}^{2^+.5}_{\mathcal{A}} - \mathsf{Adv}^{3}_{\mathcal{A}}| \leq \epsilon$.

From the above equations, we have that $|\mathsf{Adv}^2_{\mathcal{A}} - \mathsf{Adv}^3_{\mathcal{A}}| \leq (6 + 5q^{c+1})\epsilon$, as desired.

We now proceed with the formal description of the intermediate hybrids.

$\mathsf{Hybrid}_{2.\vec{j}.1}$ for $\vec{j} \in [q]^{c+1}$:

1. Adversary submits q number of message queries to the challenger. Denote $(x^j_{i,0}, x^j_{i,1})$ to be the j^{th} message query corresponding to the i^{th} position, where $j \in [q], i \in [c+1]$.
2. Challenger executes $\mathsf{MIFE}_c.\mathsf{Setup}(1^\lambda)$ to obtain $\mathsf{MIFE}_c.\mathsf{msk}$. It then executes $\mathsf{FE.Setup}(1^\lambda)$ to obtain $(\mathsf{FE.pk}, \mathsf{FE.msk})$. Finally, the challenger sets $\mathsf{MIFE}_{c+1}.\mathsf{msk}$ to be $(\mathsf{MIFE}_c.\mathsf{msk}, \mathsf{FE.pk}, \mathsf{FE.msk})$.
3. Let $\vec{j} = (j_1, \ldots, j_{c+1})$. It initializes $\mathcal{I}^*$ to be $(j_1 - 1, \ldots, j_{c+1} - 1)$.
4. It picks $\tau^*_2, \ldots, \tau^*_{c+1}$ at random with $\tau^*_i \in \{0,1\}^\lambda$.
5. The challenger then generates the challenge ciphertexts as follows. For every $i \in [c+1], j \in [q]$,
   (a) If $i = 1$ and $j \neq j_1$:
      - It draws the PRF key $K \in \{0,1\}^\lambda$ at random.
      - It then executes $\mathsf{MIFE}_c.\mathsf{KeyGen}\big(\mathsf{MIFE}_c.\mathsf{msk}, G\big)$ to obtain $\mathsf{MIFE}_c.sk_G$, where $G = \mathsf{HybGenCT}^{(c,1)}_{[x^j_{1,0}, x^j_{1,1}, j, K, \mathsf{FE.pk}, \mathcal{I}^*]}$ as defined in Figure 4.
      - It sends the challenge ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^* = \mathsf{MIFE}_c.sk_G$ to the adversary.

   (b) Else if $i = 1$ and $j = j_1$:
      - It draws the PRF key $K \in \{0,1\}^\lambda$ at random.

- Compute ct^* to be $\mathsf{FE.Enc}(\mathsf{FE.pk}, (x_{1,0}^{j_1}, \ldots, x_{c+1,0}^{j_{c+1}}); \mathsf{PRF}(K, \tau_2^* || \cdots || \tau_{c+1}^*))$.
- Puncture the PRF key K at the point $(\tau_2^* || \cdots || \tau_{c+1}^*)$ to obtain K^*.
- Execute $\mathsf{MIFE}_c.\mathsf{KeyGen}(\mathsf{MIFE}_c.\mathsf{msk}, G)$ to obtain $\mathsf{MIFE}_c.sk_G$, where $G = \mathsf{HybGenCT}^{(c,2)}_{[x_{1,0}^j, x_{1,1}^j, j, K^*, \mathsf{FE.pk}^*, \mathcal{I}^*, \mathsf{ct}^*]}$) is defined in Figure 5.
- It sends the challenge ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^* = \mathsf{MIFE}_c.sk_G$ to the adversary.

(c) Else if $2 \leq i \leq c+1$ and $j \neq j_i$:

- It picks τ at random[10].
- If $x_{i,0}^j$ is of the form $(x_1, x_2, j, \tau, i-1)$ then it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x_1, x_2, j, \tau, i), i)$ to obtain $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$. Otherwise, it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x_{i,0}^j, x_{i,1}^j, j, \tau, i), i)$ to obtain the ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$, which is sent to the adversary.

(d) Else if $2 \leq i \leq c+1$ and $j = j_i$:

- If $x_{i,0}^j$ is of the form $(x_1, x_2, 1, \tau, i-1)$ then it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x_1, x_2, 1, \tau, i), i)$ to obtain $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$. Otherwise, it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x_{i,0}^j, x_{i,1}^j, j, \tau_i^*, i), i)$ to obtain the ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$, which is sent to the adversary.

6. For every function query f, the challenger first executes $\mathsf{FE.KeyGen}(\mathsf{FE.msk}, f)$ to obtain $\mathsf{FE}.sk_f$. It then sends the functional key $\mathsf{MIFE}_{c+1}.sk_f = \mathsf{FE}.sk_f$ to the adversary.

7. Adversary outputs b'.

$\mathsf{HybGenCT}^{(c,2)}_{[x_1, x_1', i_1, K^*, \mathsf{FE.pk}^*, \mathcal{I}, \mathsf{ct}^*]}$

Input: $\Big((x_2, x_2', j_2, \tau_2, i_2), \ldots, (x_{c+1}, x_{c+1}', j_{c+1}, \tau_{c+1}, i_{c+1})\Big)$.

- $\forall k \in \{2, \ldots, c+1\}$, if $i_k \neq k$, then output $\perp$
- $\mathcal{I}' = (j_1 - 1, \ldots, j_{c+1} - 1)$
- If $\mathsf{LessThan}(\mathcal{I}', \mathcal{I}, c+1, q) = 1$ then // *$\mathcal{I}'$ is strictly less than $\mathcal{I}$*
  - $\mathsf{FE.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.pk}, (x_1', \ldots, x_{c+1}'); \mathsf{PRF}(K^*, \tau_2 || \cdots || \tau_{c+1}))$
- Else if $\mathsf{LessThan}(\mathcal{I}, \mathcal{I}', c+1, q) = 1$ then // *$\mathcal{I}'$ is strictly greater than $\mathcal{I}$*
  - $\mathsf{FE.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.pk}, (x_1, \ldots, x_{c+1}); \mathsf{PRF}(K^*, \tau_2 || \cdots || \tau_{c+1}))$
- Else,
  - $\mathsf{FE.ct} = \mathsf{ct}^*$.
- Output $\mathsf{FE.ct}$.

Figure 5

We use $\overrightarrow{1}$ to denote a vector $(1, \ldots, 1)$ of length $c+1$.

[10]The space from which τ is picked is $\{0,1\}^\lambda \backslash \tau_i^*$. Henceforth, we will not mention this explicitly.

Lemma 4. *For any PPT adversary $\mathcal{A}$, we have $|\mathsf{Adv}^2_{\mathcal{A}} - \mathsf{Adv}^{2,\overrightarrow{1}.1}_{\mathcal{A}}| \leq \epsilon$ assuming the ϵ-selective function privacy of MIFE_c.*

Proof. We describe a reduction $\mathcal{B}$ that internally uses $\mathcal{A}$ to break the function privacy property of MIFE_c. Further, we argue that advantage of $\mathcal{B}$ is $|\mathsf{Adv}^2_{\mathcal{A}} - \mathsf{Adv}^{2,\overrightarrow{1}.1}_{\mathcal{A}}|$.

The reduction $\mathcal{B}$ first executes $\mathsf{FE.Setup}(1^\lambda)$ to obtain $(\mathsf{FE.pk}, \mathsf{FE.msk})$. It also picks $(\tau^*_1, \ldots, \tau^*_{c+1})$ at random with $\tau^*_i \in \{0,1\}^\lambda$. It initializes $\mathcal{I}^*$ to be $(0, \ldots, 0)$. It initializes $(j_1, \ldots, j_{c+1})$ to be $(0, \ldots, 0)$

Upon receiving the message queries, $(x^j_{i,0}, x^j_{i,1})$, for $i \in [c+1], j \in [q]$, from the adversary $\mathcal{A}$, the reduction $\mathcal{B}$ generates the challenge ciphertexts depending on the following cases. We note that $\mathcal{B}$ makes all the queries to the challenger with respect to $i \neq 1$ (which are challenge message queries) in the beginning and only after that it makes the queries with respect to $i = 1$ (which are function queries).

- If $2 \leq i \leq c+1$ then: It first picks $\tau_{i,j}$ at random. It then sends the message pairs $(x^{i,j}_0, x^{i,j}_1)$ to the challenger, for all $2 \leq i \leq c+1, j \in [q]$, where $\big(x^{i,j}_0 = (x^j_{i,0}, x^j_{i,1}, j, \tau_{i,j}, i), x^{i,j}_1 = (x^j_{i,0}, x^j_{i,1}, j, \tau_{i,j}, i)\big)$ if $j \neq j_i$ and $\big(x^{i,j}_0 = (x^j_{i,0}, x^j_{i,1}, j, \tau^*_i, i), x^{i,j}_1 = (x^j_{i,0}, x^j_{i,1}, j, \tau^*_i, i)\big)$ if $j = j_i$. In return it receives $\mathsf{MIFE}_c.\mathsf{ct}^*_{i,j}$. It then sends $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*_{i,j}$ is set to be $\mathsf{MIFE}_c.\mathsf{ct}^*_{i,j}$ to the adversary.

- Else if $i = 1$ and $j_1 \neq 1$ then: It draws the PRF key $K \in \{0,1\}^\lambda$ at random. It then sends the function pair query $(\mathsf{HybGenCT}^{(c,1)}_{[x^j_{1,0}, x^j_{1,1}, j, K, \mathsf{FE.pk}, \mathcal{I}^*]}, \mathsf{HybGenCT}^{(c,1)}_{[x^j_{1,0}, x^j_{1,1}, j, K, \mathsf{FE.pk}, \mathcal{I}^*]})$ to the challenger of the security game of MIFE_c. In return it receives $\mathsf{MIFE}_c.sk$. It sends $\mathsf{MIFE}_{c+1}.\mathsf{ct}^* = \mathsf{MIFE}_c.sk$ to the adversary $\mathcal{A}$.

- Else if $i = 1$ and $j = j_1$ then: It draws the PRF key $K \in \{0,1\}^\lambda$ at random. It then computes ct^* to be $\mathsf{FE.Enc}(\mathsf{FE.pk}, (x^{j_1}_{1,0}, \ldots, x^{j_{c+1}}_{c+1,0}); \mathsf{PRF}(K, \tau^*_2 || \cdots || \tau^*_{c+1}))$. Further, it punctures the PRF key K at the point $(\tau^*_2 || \cdots || \tau^*_{c+1})$ to obtain K^*. It then submits the function query $(\mathsf{HybGenCT}^{(c,1)}_{[x^j_{1,0}, x^j_{1,1}, j, K, \mathsf{FE.pk}, \mathcal{I}^*]}, \mathsf{HybGenCT}^{(c,2)}_{[x^j_{1,0}, x^j_{1,1}, j, K^*, \mathsf{FE.pk}, \mathcal{I}^*, \mathsf{ct}^*]})$ to the challenger of MIFE_c. In return it receives $\mathsf{MIFE}_c.sk$. It sends the challenge ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^* = \mathsf{MIFE}_c.sk_{\mathsf{HybGenCT}^{(c,2)}}$ to the adversary.

The function queries made by the adversary are handled by the challenger as in Hybrid_2.

We claim that $\mathcal{B}$ is a valid adversary in the function privacy game of MIFE_c. To show this, note that it suffices to show the following. We denote C_0 to be $\mathsf{HybGenCT}^{(c,1)}_{[x^j_{1,0}, x^{j_1}_{1,1}, j_1, K, \mathsf{FE.pk}, \mathcal{I}^*]}$ and C_1 to be $\mathsf{HybGenCT}^{(c,2)}_{[x^j_{1,0}, x^j_{1,1}, j_1, K^*, \mathsf{FE.pk}, \mathcal{I}^*, \mathsf{ct}^*]}$, where $j_1 \in [q]$. We claim that, for all $j_2, \ldots, j_{c+1} \in [q]$,

$$C_0\Big((x^{j_2}_{2,0}, x^{j_2}_{2,1}, j_2, \tau^{j_2}_2, 2), \ldots, (x^{j_{c+1}}_{c+1,0}, x^{j_{c+1}}_{c+1,1}, 1, \tau^{j_{c+1}}_{c+1}, c+1)\Big)$$

$$= C_1\Big((x^{j_2}_{2,0}, x^{j_2}_{2,1}, j_2, \tau^{j_2}_2, 2), \ldots, (x^{j_{c+1}}_{c+1,0}, x^{j_{c+1}}_{c+1,1}, 1, \tau^{j_{c+1}}_{c+1}, c+1)\Big),$$

where $\tau^{j_i}_i$ is the random string (which is τ^*_i, when $j_i = 1$) when generating the challenge ciphertext for the j^{th} ciphertext in the i^{th} position. To show this, it suffices to just consider the input $M = \big((x^1_{2,0}, x^1_{2,1}, 1, \tau^*_1, 2), \ldots, (x^1_{c+1,0}, x^1_{c+1,1}, 1, \tau^*_{c+1}, c+1)\big)$ – on all other inputs both the circuits

behave identically. On this input M, note that (i) when $j_1 = 1$, C_0 behaves identically as C_1 and, (ii) when $j_1 = 1$, the output of C_0 is $\mathsf{FE.Enc}(\mathsf{FE.pk}, (x^1_{1,0}, \ldots, x^1_{c+1,0}))$ which is the same as $\mathsf{FE.ct}^*$, which is the output of C_1. This shows that $\mathcal{B}$ is a valid adversary.

On receiving the function query (C_0, C_1), if the challenger sent back a functional key of C_0 then we are in Hybrid_2, else if it sent a key of C_1 then we are in $\mathsf{Hybrid}_{2.\overrightarrow{1}.1}$. This completes the proof of the lemma. □

$\mathsf{Hybrid}_{2.\overrightarrow{j}.2}$ for $\overrightarrow{j} \in [q]^{c+1}$:

1. Adversary submits q number of message queries to the challenger. Denote $(x^j_{i,0}, x^j_{i,1})$ to be the j^{th} message query corresponding to the i^{th} position, where $j \in [q], i \in [c+1]$.
2. Challenger executes $\mathsf{MIFE}_c.\mathsf{Setup}(1^\lambda)$ to obtain $\mathsf{MIFE}_c.\mathsf{msk}$. It then executes $\mathsf{FE.Setup}(1^\lambda)$ to obtain $(\mathsf{FE.pk}, \mathsf{FE.msk})$. Finally, the challenger sets $\mathsf{MIFE}_{c+1}.\mathsf{msk}$ to be $(\mathsf{MIFE}_c.\mathsf{msk}, \mathsf{FE.pk}, \mathsf{FE.msk})$.
3. Let $\overrightarrow{j} = (j_1, \ldots, j_{c+1})$. It initializes $\mathcal{I}^*$ to be $(j_1 - 1, \ldots, j_{c+1} - 1)$.
4. It picks $\tau^*_2, \ldots, \tau^*_{c+1}$ at random with $\tau^*_i \in \{0,1\}^\lambda$.
5. The challenger then generates the challenge ciphertexts as follows. For every $i \in [c+1], j \in [q]$,
   (a) If $i = 1$ and $j \neq j_1$:
      - It draws the PRF key $K \in \{0,1\}^\lambda$ at random.
      - It then executes $\mathsf{MIFE}_c.\mathsf{KeyGen}(\mathsf{MIFE}_c.\mathsf{msk}, G)$ to obtain $\mathsf{MIFE}_c.sk_G$, where $G = \mathsf{HybGenCT}^{(c,1)}_{[x^j_{1,0}, x^j_{1,1}, j, K, \mathsf{FE.pk}, \mathcal{I}^*]}$ is defined in Figure 4.
      - It sends the challenge ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^* = \mathsf{MIFE}_c.sk_G$ to the adversary.
   (b) Else if $i = 1$ and $j = j_1$:
      - It draws the PRF key $K \in \{0,1\}^\lambda$ at random.
      - Compute ct^* to be $\mathsf{FE.Enc}(\mathsf{FE.pk}, (x^{j_1}_{1,0}, \ldots, x^{j_{c+1}}_{c+1,0}); R^*)$, where R^* is picked at random.
      - Puncture the PRF key K at the point $(\tau^*_2 || \cdots || \tau^*_{c+1})$ to obtain K^*.
      - Execute $\mathsf{MIFE}_c.\mathsf{KeyGen}(\mathsf{MIFE}_c.\mathsf{msk}, G)$ to obtain $\mathsf{MIFE}_c.sk_G$, where $G = \mathsf{HybGenCT}^{(c,2)}_{[x^j_{1,0}, x^j_{1,1}, j, K^*, \mathsf{FE.pk}^*, \mathcal{I}^*, \mathsf{ct}^*]}$ is defined in Figure 5.
      - It sends the challenge ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^* = \mathsf{MIFE}_c.sk_G$ to the adversary.
   (c) Else if $2 \leq i \leq c+1$ and $j \neq j_i$:
      - It picks τ at random.
      - If $x^j_{i,0}$ is of the form $(x^*_0, x^*_1, j, \tau, i-1)$ then it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x^*_0, x^*_1, j, \tau, i), i)$ to obtain $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$. Otherwise, it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x^j_{i,0}, x^j_{i,1}, j, \tau, i), i)$ to obtain the ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$, which is sent to the adversary.
   (d) Else if $2 \leq i \leq c+1$ and $j = j_i$:
      - If $x^j_{i,0}$ is of the form $(x^*_0, x^*_1, j, \tau, i-1)$ then it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x^*_0, x^*_1, j, \tau, i), i)$ to obtain $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$. Otherwise, it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x^j_{i,0}, x^j_{i,1}, j, \tau^*_i, i), i)$ to obtain the ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$, which is sent to the adversary.

6. For every function query f, the challenger first executes $\mathsf{FE.KeyGen}(\mathsf{FE.msk}, f)$ to obtain $\mathsf{FE}.sk_f$. It then sends the functional key $\mathsf{MIFE}_{c+1}.sk_f = \mathsf{FE}.sk_f$ to the adversary.

7. Adversary outputs b'.

Lemma 5. *For any PPT adversary $\mathcal{A}$, we have $|\mathsf{Adv}^{2.\overrightarrow{j}.1}_{\mathcal{A}} - \mathsf{Adv}^{2.\overrightarrow{j}.2}_{\mathcal{A}}| \leq \epsilon$ assuming the ϵ-security of F.*

Proof. We describe a reduction $\mathcal{B}$ that internally uses $\mathcal{A}$ to break the security of puncturable pseudorandom function family F. Further, we argue that advantage of $\mathcal{B}$ is $|\mathsf{Adv}^{2}_{\mathcal{A}} - \mathsf{Adv}^{2.\overrightarrow{1}.1}_{\mathcal{A}}|$.

The reduction $\mathcal{B}$ picks $(\tau^*_2, \ldots, \tau^*_{c+1})$ at random. It then queries the PRF oracle with $(\tau^*_2, \ldots, \tau^*_{c+1})$ to get R^* and a punctured PRF key K^* that is punctured at the point $(\tau^*_2, \ldots, \tau^*_{c+1})$. The reduction algorithm, $\mathcal{B}$ then proceeds as in $\mathsf{Hybrid}_{2.\overrightarrow{j}.2}$, where it uses the randomness R^* in the generation of the ciphertext $\mathsf{FE.ct}^*$.

If the PRF oracle generates R^* to be the output of PRF evaluation on the point $(\tau^*_2, \ldots, \tau^*_{c+1})$, then we are in $\mathsf{Hybrid}_{2.\overrightarrow{j}.1}$, else if R^* is picked at random then we are in $\mathsf{Hybrid}_{2.\overrightarrow{j}.2}$. This completes the proof of the lemma. □

$\mathsf{Hybrid}_{2.\overrightarrow{j}.3}$ for $\overrightarrow{j} \in [q]^{c+1}$:

1. Adversary submits q number of message queries to the challenger. Denote $(x^j_{i,0}, x^j_{i,1})$ to be the j^{th} message query corresponding to the i^{th} position, where $j \in [q], i \in [c+1]$.

2. Challenger executes $\mathsf{MIFE}_c.\mathsf{Setup}(1^\lambda)$ to obtain $\mathsf{MIFE}_c.\mathsf{msk}$. It then executes $\mathsf{FE.Setup}(1^\lambda)$ to obtain $(\mathsf{FE.pk}, \mathsf{FE.msk})$. Finally, the challenger sets $\mathsf{MIFE}_{c+1}.\mathsf{msk}$ to be $(\mathsf{MIFE}_c.\mathsf{msk}, \mathsf{FE.pk}, \mathsf{FE.msk})$.

3. Let $\overrightarrow{j} = (j_1, \ldots, j_{c+1})$. It initializes $\mathcal{I}^*$ to be $(j_1 - 1, \ldots, j_{c+1} - 1)$.

4. It picks $\tau^*_2, \ldots, \tau^*_{c+1}$ at random with $\tau^*_i \in \{0,1\}^\lambda$.

5. The challenger then generates the challenge ciphertexts as follows. For every $i \in [c+1], j \in [q]$,

   (a) $i = 1$ and $j \neq j_1$:
   - It draws the PRF key $K \in \{0,1\}^\lambda$ at random.
   - It then executes $\mathsf{MIFE}_c.\mathsf{KeyGen}(\mathsf{MIFE}_c.\mathsf{msk}, G)$ to obtain $\mathsf{MIFE}_c.sk_G$, where $\mathsf{HybGenCT}^{(c,1)}_{[x^j_{1,0}, x^j_{1,1}, j, K, \mathsf{FE.pk}, \mathcal{I}^*]}$ is defined in Figure 4.
   - It sends the challenge ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^* = \mathsf{MIFE}_c.sk_G$ to the adversary.

   (b) $i = 1$ and $j = j_1$:
   - It draws the PRF key $K \in \{0,1\}^\lambda$ at random.
   - Compute ct^* to be $\mathsf{FE.Enc}(\mathsf{FE.pk}, (x^{j_1}_{1,1}, \ldots, x^{j_{c+1}}_{c+1,1}); R^*)$, where R^* is picked at random.
   - Puncture the PRF key K at the point $(\tau^*_2 || \cdots || \tau^*_{c+1})$ to obtain K^*.
   - Execute $\mathsf{MIFE}_c.\mathsf{KeyGen}(\mathsf{MIFE}_c.\mathsf{msk}, G)$ to obtain $\mathsf{MIFE}_c.sk_G$, where $G = \mathsf{HybGenCT}^{(c,2)}_{[x^j_{1,0}, x^j_{1,1}, j, K^*, \mathsf{FE.pk}^*, \mathcal{I}^*, \mathsf{ct}^*]}$ is defined in Figure 5.

- It sends the challenge ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^* = \mathsf{MIFE}_c.sk_G$ to the adversary.

(c) $2 \leq i \leq c+1$ and $j \neq j_i$:

- It picks τ at random.
- If $x^j_{i,0}$ is of the form $(x^*_0, x^*_1, j, \tau, i-1)$ then it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x^*_0, x^*_1, j, \tau, i), i)$ to obtain $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$. Otherwise, it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x^j_{i,0}, x^j_{i,1}, j, \tau, i), i)$ to obtain the ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$, which is sent to the adversary.

(d) $2 \leq i \leq c+1$ and $j = j_i$:

- If $x^j_{i,0}$ is of the form $(x^*_0, x^*_1, j, \tau, i-1)$ then it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x^*_0, x^*_1, j, \tau, i), i)$ to obtain $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$. Otherwise, it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x^j_{i,0}, x^j_{i,1}, j, \tau^*_i, i), i)$ to obtain the ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$, which is sent to the adversary.

6. For every function query f, the challenger first executes $\mathsf{FE}.\mathsf{KeyGen}(\mathsf{FE}.\mathsf{msk}, f)$ to obtain $\mathsf{FE}.sk_f$. It then sends the functional key $\mathsf{MIFE}_{c+1}.sk_f = \mathsf{FE}.sk_f$ to the adversary.

7. Adversary outputs b'.

Lemma 6. *For any PPT adversary $\mathcal{A}$, we have $|\mathsf{Adv}^{2.\overrightarrow{j}.2}_{\mathcal{A}} - \mathsf{Adv}^{2.\overrightarrow{j}.3}_{\mathcal{A}}| \leq \epsilon$ assuming the ϵ-selective security of FE.*

Proof. We design a reduction, that uses $\mathcal{A}$, to break the security of FE with advantage ϵ, where $\epsilon = |\mathsf{Adv}^{2.\overrightarrow{j}.2}_{\mathcal{A}} - \mathsf{Adv}^{2.\overrightarrow{j}.3}_{\mathcal{A}}|$.

The reduction $\mathcal{B}$ answers the message queries made by the adversary $\mathcal{A}$ as in $\mathsf{Hybrid}_{2.\overrightarrow{j}.2}$. The only difference is that, the ciphertext $\mathsf{FE}.\mathsf{ct}^*$, which is hardwired in $\mathsf{HybGenCT}^{(c,2)}$, is not generated by $\mathcal{B}$ but instead is obtained from the challenger of FE by submitting the message query $\big((x^{j_1}_{1,0}, \ldots, x^{j_{c+1}}_{c+1,0}), (x^{j_1}_{1,1}, \ldots, x^{j_{c+1}}_{c+1,1})\big)$. The challenger also sends the public key $\mathsf{FE}.\mathsf{pk}$ which is used by the reduction to generate the challenge ciphertexts.

Upon receiving any function query f from the adversary $\mathcal{A}$, the reduction $\mathcal{B}$ forwards the function f to the challenger of FE. In return it receives $\mathsf{FE}.sk_f$. $\mathcal{B}$ then sends $\mathsf{MIFE}_{c+1}.sk = \mathsf{FE}.sk_f$ to $\mathcal{A}$.

We argue that $\mathcal{B}$ is a valid adversary in the security game of FE. To see this it suffices to argue that for every function query f made by $\mathcal{B}$, $f(x^{j_1}_{1,0}, \ldots, x^{j_{c+1}}_{c+1,0}) = f(x^{j_1}_{1,1}, \ldots, x^{j_{c+1}}_{c+1,1})$, This can be asserted from the fact that $\mathcal{A}$ is a valid attacker in the security game of MIFE_{c+1}.

If the challenger of FE encrypts the message $(x^{j_1}_{1,0}, \ldots, x^{j_{c+1}}_{c+1,0})$ then we are in $\mathsf{Hybrid}_{2.\overrightarrow{j}.2}$ and if it encrypts the message $(x^{j_1}_{1,1}, \ldots, x^{j_{c+1}}_{c+1,1})$ then we are in $\mathsf{Hybrid}_{2.\overrightarrow{j}.3}$. This completes the proof of the lemma. □

$\mathsf{Hybrid}_{2.\overrightarrow{j}.4}$ for $\overrightarrow{j} \in [q]^{c+1}$:

1. Adversary submits q number of message queries to the challenger. Denote $(x^j_{i,0}, x^j_{i,1})$ to be the j^{th} message query corresponding to the i^{th} position, where $j \in [q], i \in [c+1]$.

2. Challenger executes $\mathsf{MIFE}_c.\mathsf{Setup}(1^\lambda)$ to obtain $\mathsf{MIFE}_c.\mathsf{msk}$. It then executes $\mathsf{FE}.\mathsf{Setup}(1^\lambda)$ to obtain $(\mathsf{FE}.\mathsf{pk}, \mathsf{FE}.\mathsf{msk})$. Finally, the challenger sets $\mathsf{MIFE}_{c+1}.\mathsf{msk}$ to be $(\mathsf{MIFE}_c.\mathsf{msk}, \mathsf{FE}.\mathsf{pk}, \mathsf{FE}.\mathsf{msk})$.

3. Let $\overrightarrow{j} = (j_1, \ldots, j_{c+1})$. It initializes $\mathcal{I}^*$ to be $(j_1 - 1, \ldots, j_{c+1} - 1)$.

4. It picks $\tau_2^*, \ldots, \tau_{c+1}^*$ at random with $\tau_i^* \in \{0,1\}^\lambda$.

5. The challenger then generates the challenge ciphertexts as follows. For every $i \in [c+1], j \in [q]$,

   (a) $i = 1$ and $j \neq j_1$:
   - It draws the PRF key $K \in \{0,1\}^\lambda$ at random.
   - It then executes $\mathsf{MIFE}_c.\mathsf{KeyGen}(\mathsf{MIFE}_c.\mathsf{msk}, G)$ to obtain $\mathsf{MIFE}_c.sk_G$, where $G = \mathsf{HybGenCT}^{(c,1)}_{[x^j_{1,0}, x^j_{1,1}, j, K, \mathsf{FE.pk}, \mathcal{I}^*]}$ is defined in Figure 4.
   - It sends the challenge ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^* = \mathsf{MIFE}_c.sk_G$ to the adversary.

   (b) $i = 1$ and $j = j_1$:
   - It draws the PRF key $K \in \{0,1\}^\lambda$ at random.
   - Compute ct^* to be $\mathsf{FE.Enc}(\mathsf{FE.pk}, (x^{j_1}_{1,1}, \ldots, x^{j_{c+1}}_{c+1,1}); \mathsf{PRF}(K, \tau_2^* || \cdots || \tau_{c+1}^*))$.
   - Puncture the PRF key K at the point $(\tau_2^* || \cdots || \tau_{c+1}^*)$ to obtain K^*.
   - Execute $\mathsf{MIFE}_c.\mathsf{KeyGen}(\mathsf{MIFE}_c.\mathsf{msk}, G)$ to obtain $\mathsf{MIFE}_c.sk_G$, where $G = \mathsf{HybGenCT}^{(c,2)}_{[x^j_{1,0}, x^j_{1,1}, j, K^*, \mathsf{FE.pk}^*, \mathcal{I}^*, \mathsf{ct}^*]}$ is defined in Figure 5.
   - It sends the challenge ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^* = \mathsf{MIFE}_c.sk_G$ to the adversary.

   (c) $2 \leq i \leq c+1$ and $j \neq j_i$:
   - It picks τ at random.
   - If $x^j_{i,0}$ is of the form $(x_0^*, x_1^*, j, \tau, i-1)$ then it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x_0^*, x_1^*, j, \tau, i), i)$ to obtain $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$. Otherwise, it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x^j_{i,0}, x^j_{i,1}, j, \tau, i), i)$ to obtain the ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$, which is sent to the adversary.

   (d) $2 \leq i \leq c+1$ and $j = j_i$:
   - If $x^j_{i,0}$ is of the form $(x_0^*, x_1^*, j, \tau, i-1)$ then it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x_0^*, x_1^*, j, \tau, i), i)$ to obtain $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$. Otherwise, it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x^j_{i,0}, x^j_{i,1}, j, \tau_i^*, i), i)$ to obtain the ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$, which is sent to the adversary.

6. For every function query f, the challenger first executes $\mathsf{FE.KeyGen}(\mathsf{FE.msk}, f)$ to obtain $\mathsf{FE}.sk_f$. It then sends the functional key $\mathsf{MIFE}_{c+1}.sk_f = \mathsf{FE}.sk_f$ to the adversary.

7. Adversary outputs b'.

Lemma 7. *For any PPT adversary $\mathcal{A}$, we have $|\mathsf{Adv}^{2.\vec{j}.3}_{\mathcal{A}} - \mathsf{Adv}^{2.\vec{j}.4}_{\mathcal{A}}| \leq \epsilon$ assuming the ϵ-security of puncturable pseudorandom functions F.*

The proof of the above lemma is similar to the proof of Lemma 5.

$\mathsf{Hybrid}_{2.\vec{j}.5}$ for $\vec{j} \in [q]^{c+1}$:

1. Adversary submits q number of message queries to the challenger. Denote $(x^j_{i,0}, x^j_{i,1})$ to be the j^{th} message query corresponding to the i^{th} position, where $j \in [q], i \in [c+1]$.

2. Challenger executes $\mathsf{MIFE}_c.\mathsf{Setup}(1^\lambda)$ to obtain $\mathsf{MIFE}_c.\mathsf{msk}$. It then executes $\mathsf{FE.Setup}(1^\lambda)$ to obtain $(\mathsf{FE.pk}, \mathsf{FE.msk})$. Finally, the challenger sets $\mathsf{MIFE}_{c+1}.\mathsf{msk}$ to be $(\mathsf{MIFE}_c.\mathsf{msk}, \mathsf{FE.pk}, \mathsf{FE.msk})$.

3. Let $\overrightarrow{j} = (j_1, \ldots, j_{c+1})$. Let $\mathcal{I} = (j_1 - 1, \ldots, j_{c+1} - 1)$. It initializes $\mathcal{I}^*$ to be $\mathcal{I} + (0, \ldots, 0, 1)$ (here the addition operation is performed in $\mathbb{Z}_q^{c+1}$).

4. The challenger then generates the challenge ciphertexts as follows. For every $i \in [c+1], j \in [q]$,

   (a) $i = 1$:
   - It draws the PRF key $K \in \{0,1\}^\lambda$ at random.
   - It then executes $\mathsf{MIFE}_c.\mathsf{KeyGen}\big(\mathsf{MIFE}_c.\mathsf{msk}, G\big)$ to obtain $\mathsf{MIFE}_c.sk_G$, where $G = \mathsf{HybGenCT}^{(c,1)}_{[x^j_{1,0}, x^j_{1,1}, j, K, \mathsf{FE.pk}, \mathcal{I}^*]}$ is defined in Figure 4.
   - It sends the challenge ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^* = \mathsf{MIFE}_c.sk_G$ to the adversary.

   (b) $2 \leq i \leq c+1$:
   - It picks τ at random.
   - If $x^j_{i,0}$ is of the form $(x^*_0, x^*_1, j, \tau, i-1)$ then it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x^*_0, x^*_1, j, \tau, i), i)$ to obtain $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$. Otherwise, it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x^j_{i,0}, x^j_{i,1}, j, \tau, i), i)$ to obtain the ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$, which is sent to the adversary.

5. For every function query f, the challenger first executes $\mathsf{FE.KeyGen}(\mathsf{FE.msk}, f)$ to obtain $\mathsf{FE}.sk_f$. It then sends the functional key $\mathsf{MIFE}_{c+1}.sk_f = \mathsf{FE}.sk_f$ to the adversary.

6. Adversary outputs b'.

Lemma 8. *For any PPT adversary $\mathcal{A}$, we have $|\mathsf{Adv}^{2,\overrightarrow{j},4}_{\mathcal{A}} - \mathsf{Adv}^{2,\overrightarrow{j},5}_{\mathcal{A}}| \leq \epsilon$ assuming the ϵ-selective function privacy of MIFE_c.*

Proof. We design a reduction that uses $\mathcal{A}$ to break the selective function privacy game of MIFE_c.

The reduction $\mathcal{B}$ interacts with the adversary $\mathcal{A}$ by simulating the role of the challenger in the game of MIFE_c. Upon receiving the message queries, $(x^j_{i,0}, x^j_{i,1})$, for all $i \in [c+1]$, $j \in [q]$, from the adversary $\mathcal{A}$, the $\mathcal{B}$ does the following. The reduction first executes $\mathsf{FE.Setup}(1^\lambda)$ to obtain $(\mathsf{FE.pk}, \mathsf{FE.msk})$. Suppose $\overrightarrow{j} = (j_1, \ldots, j_{c+1})$ and let $\mathcal{I} = (j_1 - 1, \ldots, j_{c+1} - 1)$. $\mathcal{B}$ then initializes $\mathcal{I}^*$ to be $\mathcal{I} + 1$ (here the addition operation is performed in $\mathbb{Z}_q^{c+1}$). The reduction $\mathcal{B}$ then generates the challenge ciphertexts according to the following cases. We note that the reduction $\mathcal{B}$ makes all the queries to the challenger with respect to $i \neq 1$ in the beginning (which are message queries) and only after that it makes all the queries to the challenger with respect to $= 1$ (function queries).

1. $i = 1$ and $j \neq j_1$: The reduction $\mathcal{B}$ first draws the PRF key $K \in \{0,1\}^\lambda$ at random. It then submits the query (C, C) to the challenger of the MIFE_c game, where $C = \mathsf{HybGenCT}^{(c,1)}_{[x^j_{1,0}, x^j_{1,1}, j, K, \mathsf{FE.pk}, \mathcal{I}^*]}$. In return it receives $\mathsf{MIFE}_c.sk$. The reduction then sends $\mathsf{MIFE}_{c+1}.\mathsf{ct}^* = \mathsf{MIFE}_c.sk$ to $\mathcal{A}$.

2. $i = 1$ and $j = j_1$: The reduction $\mathcal{B}$ draws the PRF key $K \in \{0,1\}^\lambda$ at random. It then computes ct^* to be $\mathsf{FE.Enc}(\mathsf{FE.pk}, (x^{j_1}_{1,1}, \ldots, x^{j_{c+1}}_{c+1,1}); R)$, where $R = \mathsf{PRF}(K, \tau^*_2 || \cdots || \tau^*_{c+1}$. It then punctures the PRF key K at the point $(\tau^*_2 || \cdots || \tau^*_{c+1})$ to obtain K^*. It then submits the function query (C_0, C_1) to the challenger, where $C_0 = \mathsf{HybGenCT}^{(c,2)}_{[x^j_{1,0}, x^j_{1,1}, j, K^*, \mathsf{FE.pk}^*, \mathcal{I}^*, \mathsf{ct}^*]}$ and $C_1 = \mathsf{HybGenCT}^{(c,1)}_{[x^j_{1,0}, x^j_{1,1}, j, K, \mathsf{FE.pk}, \mathcal{I}^*]}$. In return it receives $\mathsf{MIFE}_c.sk$. It then sends $\mathsf{MIFE}_{c+1}.\mathsf{ct}^* = \mathsf{MIFE}_c.sk$ to the adversary $\mathcal{A}$.

3. $2 \leq i \leq c+1$ and $j \neq j_i$: The reduction $\mathcal{B}$ picks τ at random. It then submits the message query (M, M) to the challenger, where $M = (x^j_{i,0}, x^j_{i,1}, j, \tau, i)$. In response, it obtains $\mathsf{MIFE}_c.\mathsf{ct}^*$. The reduction $\mathcal{B}$ then sends $\mathsf{MIFE}_{c+1}.\mathsf{ct}^* = \mathsf{MIFE}_c.\mathsf{ct}^*$ to the adversary.

4. $2 \leq i \leq c+1$ and $j = j_i$: The reduction $\mathcal{B}$ submits the message query (M, M) to the challenger, where $M = (x^j_{i,0}, x^j_{i,1}, j, \tau, i)$. In response, it obtains $\mathsf{MIFE}_c.\mathsf{ct}^*$. The reduction $\mathcal{B}$ then sends $\mathsf{MIFE}_{c+1}.\mathsf{ct}^* = \mathsf{MIFE}_c.\mathsf{ct}^*$ to the adversary.

$\mathcal{B}$ then answers the function queries as in $\mathsf{Hybrid}_{2.\overrightarrow{j}.4}$ (or $\mathsf{Hybrid}_{2.\overrightarrow{j}.5}$).

We claim that $\mathcal{B}$ is a valid adversary in the function privacy game of MIFE_c. To see this, note that it suffices for us to show the following, where C_0 and C_1 are as defined above.

$$C_0\big((x^{j_1}_{1,0}, x^{j_1}_{1,1}, j_1, \tau^*_1, 1), \ldots, (x^{j_1}_{1,0}, x^{j_1}_{1,1}, j_1, \tau^*_1, 1)\big) = C_1\big((x^{j_1}_{1,0}, x^{j_1}_{1,1}, j_1, \tau^*_1, 1), \ldots, (x^{j_1}_{1,0}, x^{j_1}_{1,1}, j_1, \tau^*_1, 1)\big)$$

The output of C_0 is essentially $\mathsf{FE.ct}^*$ which is defined to be $\mathsf{FE.Enc}(\mathsf{FE.pk}, x^{j_1}_{1,1}||\cdots||x^{j_{c+1}}_{c+1,1}; R)$. Further note that the output of C_1 is also $\mathsf{FE.Enc}(\mathsf{FE.pk}, x^{j_1}_{1,1}||\cdots||x^{j_{c+1}}_{c+1,1}; R)$. In addition, $\mathcal{B}$ makes all the message queries at the beginning of the game and only it makes all the function queries. Hence, $\mathcal{B}$ is a valid adversary in the security game of MIFE_c.

If the challenger sends a functional key of C_0 then we are in $\mathsf{Hybrid}_{2.\overrightarrow{j}.4}$ and if it sends a functional key of C_1 then we are in $\mathsf{Hybrid}_{2.\overrightarrow{j}.5}$. And so, $\mathcal{B}$ breaks the function privacy property of MIFE_c with advantage $|\mathsf{Adv}^{2.\overrightarrow{j}.4}_{\mathcal{A}} - \mathsf{Adv}^{2.\overrightarrow{j}.5}_{\mathcal{A}}|$. □

Lemma 9. *For any PPT adversary $\mathcal{A}$, for all $\overrightarrow{j} \in [q]^{c+1}$ and $\overrightarrow{j} \neq (q, \ldots, q)$, we have $|\mathsf{Adv}^{2.\overrightarrow{j}.5}_{\mathcal{A}} - \mathsf{Adv}^{2.\overrightarrow{j}+1.1}_{\mathcal{A}}| \leq \epsilon$ assuming the ϵ-selective function privacy game of MIFE_c.*

The proof of the above lemma is similar to the proof of Lemma 4.

$\mathsf{Hybrid}_{2^+.1}$ for $\overrightarrow{j} \in [q]^{c+1}$:

1. Adversary submits q number of message queries to the challenger. Denote $(x^j_{i,0}, x^j_{i,1})$ to be the j^{th} message query corresponding to the i^{th} position, where $j \in [q], i \in [c+1]$.

2. Challenger executes $\mathsf{MIFE}_c.\mathsf{Setup}(1^\lambda)$ to obtain $\mathsf{MIFE}_c.\mathsf{msk}$. It then executes $\mathsf{FE.Setup}(1^\lambda)$ to obtain $(\mathsf{FE.pk}, \mathsf{FE.msk})$. Finally, the challenger sets $\mathsf{MIFE}_{c+1}.\mathsf{msk}$ to be $(\mathsf{MIFE}_c.\mathsf{msk}, \mathsf{FE.pk}, \mathsf{FE.msk})$.

3. It sets $\overrightarrow{j} = (j_1, \ldots, j_{c+1})$, with $j_i = q-1$ for all $i \in [c+1]$. It initializes $\mathcal{I}^*$ to be $(j_1 - 1, \ldots, j_{c+1} - 1)$.

4. It picks $\tau^*_2, \ldots, \tau^*_{c+1}$ at random with $\tau^*_i \in \{0,1\}^\lambda$.

5. The challenger then generates the challenge ciphertexts as follows. For every $i \in [c+1], j \in [q]$,

   (a) $i = 1$ and $j \neq j_1$:
   - It draws the PRF key $K \in \{0,1\}^\lambda$ at random.
   - It then executes $\mathsf{MIFE}_c.\mathsf{KeyGen}\big(\mathsf{MIFE}_c.\mathsf{msk}, G\big)$ to obtain $\mathsf{MIFE}_c.sk_G$, where $G = \mathsf{HybGenCT}^{(c,1)}_{[x^j_{1,0}, x^j_{1,1}, j, K, \mathsf{FE.pk}, \mathcal{I}^*]}$ is defined in Figure 4.
   - It sends the challenge ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^* = \mathsf{MIFE}_c.sk_G$ to the adversary.

(b) $i = 1$ and $j = j_1$:

- It draws the PRF key $K \in \{0,1\}^\lambda$ at random.
- Compute ct^* to be $\mathsf{FE.Enc}(\mathsf{FE.pk}, (x_{1,0}^{j_1}, \ldots, x_{c+1,0}^{j_{c+1}}); \mathsf{PRF}(K, \tau_2^* || \cdots || \tau_{c+1}^*))$.
- Puncture the PRF key K at the point $(\tau_2^* || \cdots || \tau_{c+1}^*)$ to obtain K^*.
- Execute $\mathsf{MIFE}_c.\mathsf{KeyGen}(\mathsf{MIFE}_c.\mathsf{msk}, G)$ to obtain $\mathsf{MIFE}_c.sk_G$, where $G = \mathsf{HybGenCT}^{(c,3)}_{[x_{1,0}^j, x_{1,1}^j, j, K^*, \mathsf{FE.pk}^*, \mathcal{I}^*, \mathsf{ct}^*]}$ is defined in Figure 6.
- It sends the challenge ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^* = \mathsf{MIFE}_c.sk_G$ to the adversary.

(c) $2 \le i \le c+1$ and $j \neq j_i$:

- It picks τ at random.
- If $x_{i,0}^j$ is of the form $(x_0^*, x_1^*, j, \tau, i-1)$ then it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x_0^*, x_1^*, j, \tau, i), i)$ to obtain $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$. Otherwise, it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x_{i,0}^j, x_{i,1}^j, j, \tau, i), i)$ to obtain the ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$, which is sent to the adversary.

(d) $2 \le i \le c+1$ and $j = j_i$:

- If $x_{i,0}^j$ is of the form $(x_0^*, x_1^*, j, \tau, i-1)$ then it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x_0^*, x_1^*, j, \tau, i), i)$ to obtain $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$. Otherwise, it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x_{i,0}^j, x_{i,1}^j, j, \tau_i^*, i), i)$ to obtain the ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$, which is sent to the adversary.

6. For every function query f, the challenger first executes $\mathsf{FE.KeyGen}(\mathsf{FE.msk}, f)$ to obtain $\mathsf{FE}.sk_f$. It then sends the functional key $\mathsf{MIFE}_{c+1}.sk_f = \mathsf{FE}.sk_f$ to the adversary.
7. Adversary outputs b'.

$\mathsf{HybGenCT}^{(c,3)}_{[x_1, x_1', i_1, K^*, \mathsf{FE.pk}^*, \mathcal{I}, \mathsf{ct}^*]}$

Input: $\Big((x_2, x_2', j_2, \tau_2, i_2), \ldots, (x_{c+1}, x_{c+1}', j_{c+1}, \tau_{c+1}, i_{c+1})\Big)$.

- $\forall k \in \{2, \ldots, c+1\}$, if $i_k \neq k$, then output $\perp$
- $\mathcal{I}' = (j_1 - 1, \ldots, j_n - 1)$
- If $\mathsf{LessThan}(\mathcal{I}', \mathcal{I}, c+1, q) = 1$ then
  - $\mathsf{FE.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.pk}, (x_1', \ldots, x_{c+1}'); \mathsf{PRF}(K^*, \tau_2 || \cdots || \tau_{c+1}))$
- Else if $\mathsf{LessThan}(\mathcal{I}, \mathcal{I}', c+1, q) = 1$ and $\mathcal{I} \neq (q-1, \ldots, q-1)$ then
  - $\mathsf{FE.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.pk}, (x_1, \ldots, x_{c+1}); \mathsf{PRF}(K^*, \tau_2 || \cdots || \tau_{c+1}))$
- Else,
  - $\mathsf{FE.ct} = \mathsf{ct}^*$.
- Output $\mathsf{FE.ct}$.

Figure 6

Lemma 10. *For any PPT adversary $\mathcal{A}$, we have $|\mathsf{Adv}^{2.(q,\ldots,q).5}_{\mathcal{A}} - \mathsf{Adv}^{2^+.1}_{\mathcal{A}}| \leq \epsilon$ assuming the ϵ-selective function privacy of MIFE_c.*

The proof of the above lemma follows along the same lines as the proof of Lemma 4.

$\mathsf{Hybrid}_{2^+.2}$ for $\overrightarrow{j} \in [q]^{c+1}$:

1. Adversary submits q number of message queries to the challenger. Denote $(x^j_{i,0}, x^j_{i,1})$ to be the j^{th} message query corresponding to the i^{th} position, where $j \in [q], i \in [c+1]$.
2. Challenger executes $\mathsf{MIFE}_c.\mathsf{Setup}(1^\lambda)$ to obtain $\mathsf{MIFE}_c.\mathsf{msk}$. It then executes $\mathsf{FE.Setup}(1^\lambda)$ to obtain $(\mathsf{FE.pk}, \mathsf{FE.msk})$. Finally, the challenger sets $\mathsf{MIFE}_{c+1}.\mathsf{msk}$ to be $(\mathsf{MIFE}_c.\mathsf{msk}, \mathsf{FE.pk}, \mathsf{FE.msk})$.
3. It sets $\overrightarrow{j} = (j_1, \ldots, j_{c+1})$, with $j_i = q$ for all $i \in [c+1]$. It initializes $\mathcal{I}^*$ to be $(j_1 - 1, \ldots, j_{c+1} - 1)$.
4. It picks $\tau^*_2, \ldots, \tau^*_{c+1}$ at random with $\tau^*_i \in \{0,1\}^\lambda$.
5. The challenger then generates the challenge ciphertexts as follows. For every $i \in [c+1], j \in [q]$,
   (a) $i = 1$ and $j \neq j_1$:
   - It draws the PRF key $K \in \{0,1\}^\lambda$ at random.
   - It then executes $\mathsf{MIFE}_c.\mathsf{KeyGen}(\mathsf{MIFE}_c.\mathsf{msk}, G)$ to obtain $\mathsf{MIFE}_c.sk_G$, where $G = \mathsf{HybGenCT}^{(c,1)}_{[x^j_{1,0}, x^j_{1,1}, j, K, \mathsf{FE.pk}, \mathcal{I}^*]}$ is defined in Figure 4.
   - It sends the challenge ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^* = \mathsf{MIFE}_c.sk_G$ to the adversary.

   (b) $i = 1$ and $j = j_1$:
   - It draws the PRF key $K \in \{0,1\}^\lambda$ at random.
   - Compute ct^* to be $\mathsf{FE.Enc}(\mathsf{FE.pk}, (x^{j_1}_{1,0}, \ldots, x^{j_{c+1}}_{c+1,0}); R^*)$, where R^* is picked at random .
   - Puncture the PRF key K at the point $(\tau^*_2 || \cdots || \tau^*_{c+1})$ to obtain K^*.
   - Execute $\mathsf{MIFE}_c.\mathsf{KeyGen}(\mathsf{MIFE}_c.\mathsf{msk}, G)$ to obtain $\mathsf{MIFE}_c.sk_G$, where $G = \mathsf{HybGenCT}^{(c,3)}_{[x^j_{1,0}, x^j_{1,1}, j, K^*, \mathsf{FE.pk}^*, \mathcal{I}^*, \mathsf{ct}^*]}$ is defined in Figure 6.
   - It sends the challenge ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^* = \mathsf{MIFE}_c.sk_G$ to the adversary.

   (c) $2 \leq i \leq c+1$ and $j \neq j_i$:
   - It picks τ at random.
   - If $x^j_{i,0}$ is of the form $(x^*_0, x^*_1, j, \tau, i-1)$ then it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x^*_0, x^*_1, j, \tau, i), i)$ to obtain $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$. Otherwise, it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x^j_{i,0}, x^j_{i,1}, j, \tau, i), i)$ to obtain the ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$, which is sent to the adversary.

   (d) $2 \leq i \leq c+1$ and $j = j_i$:
   - If $x^j_{i,0}$ is of the form $(x^*_0, x^*_1, j, \tau, i-1)$ then it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x^*_0, x^*_1, j, \tau, i), i)$ to obtain $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$. Otherwise, it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x^j_{i,0}, x^j_{i,1}, j, \tau^*_i, i), i)$ to obtain the ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$, which is sent to the adversary.
6. For every function query f, the challenger first executes $\mathsf{FE.KeyGen}(\mathsf{FE.msk}, f)$ to obtain $\mathsf{FE}.sk_f$. It then sends the functional key $\mathsf{MIFE}_{c+1}.sk_f = \mathsf{FE}.sk_f$ to the adversary.

7. Adversary outputs b'.

Lemma 11. *For any PPT adversary $\mathcal{A}$, we have $|\mathsf{Adv}^{2^+.2}_{\mathcal{A}} - \mathsf{Adv}^{2^+.1}_{\mathcal{A}}| \leq \epsilon$ assuming the ϵ-selective security game of F.*

The proof of the above lemma follows along the same lines as the proof of Lemma 5.
$\mathsf{Hybrid}_{2^+.3}$ for $\overrightarrow{j} \in [q]^{c+1}$:

1. Adversary submits q number of message queries to the challenger. Denote $(x^j_{i,0}, x^j_{i,1})$ to be the j^{th} message query corresponding to the i^{th} position, where $j \in [q], i \in [c+1]$.
2. Challenger executes $\mathsf{MIFE}_c.\mathsf{Setup}(1^\lambda)$ to obtain $\mathsf{MIFE}_c.\mathsf{msk}$. It then executes $\mathsf{FE.Setup}(1^\lambda)$ to obtain $(\mathsf{FE.pk}, \mathsf{FE.msk})$. Finally, the challenger sets $\mathsf{MIFE}_{c+1}.\mathsf{msk}$ to be $(\mathsf{MIFE}_c.\mathsf{msk}, \mathsf{FE.pk}, \mathsf{FE.msk})$.
3. It sets $\overrightarrow{j} = (j_1, \ldots, j_{c+1})$, with $j_i = q - 1$ for all $i \in [c+1]$. It initializes $\mathcal{I}^*$ to be $(j_1 - 1, \ldots, j_{c+1} - 1)$.
4. It picks $\tau^*_2, \ldots, \tau^*_{c+1}$ at random with $\tau^*_i \in \{0,1\}^\lambda$.
5. The challenger then generates the challenge ciphertexts as follows. For every $i \in [c+1], j \in [q]$,

   (a) $i = 1$ and $j \neq j_1$:
   - It draws the PRF key $K \in \{0,1\}^\lambda$ at random.
   - It then executes $\mathsf{MIFE}_c.\mathsf{KeyGen}\big(\mathsf{MIFE}_c.\mathsf{msk}, G\big)$ to obtain $\mathsf{MIFE}_c.sk_G$, where $G = \mathsf{HybGenCT}^{(c,1)}_{[x^j_{1,0}, x^j_{1,1}, j, K, \mathsf{FE.pk}, \mathcal{I}^*]}$ is defined in Figure 4.
   - It sends the challenge ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^* = \mathsf{MIFE}_c.sk_G$ to the adversary.

   (b) $i = 1$ and $j = j_1$:
   - It draws the PRF key $K \in \{0,1\}^\lambda$ at random.
   - Compute ct^* to be $\mathsf{FE.Enc}(\mathsf{FE.pk}, (x^{j_1}_{1,1}, \ldots, x^{j_{c+1}}_{c+1,1}); R^*)$, where R^* is picked at random.
   - Puncture the PRF key K at the point $(\tau^*_2 || \cdots || \tau^*_{c+1})$ to obtain K^*.
   - Execute $\mathsf{MIFE}_c.\mathsf{KeyGen}(\mathsf{MIFE}_c.\mathsf{msk}, G)$ to obtain $\mathsf{MIFE}_c.sk_G$, where $G = \mathsf{HybGenCT}^{(c,3)}_{[x^j_{1,0}, x^j_{1,1}, j, K^*, \mathsf{FE.pk}^*, \mathcal{I}^*, \mathsf{ct}^*]}$ is defined in Figure 6.
   - It sends the challenge ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^* = \mathsf{MIFE}_c.sk_G$ to the adversary.

   (c) $2 \leq i \leq c+1$ and $j \neq j_i$:
   - It picks τ at random.
   - If $x^j_{i,0}$ is of the form $(x^*_0, x^*_1, j, \tau, i-1)$ then it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x^*_0, x^*_1, j, \tau, i), i)$ to obtain $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$. Otherwise, it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x^j_{i,0}, x^j_{i,1}, j, \tau, i), i)$ to obtain the ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$, which is sent to the adversary.

   (d) $2 \leq i \leq c+1$ and $j = j_i$:
   - If $x^j_{i,0}$ is of the form $(x^*_0, x^*_1, j, \tau, i-1)$ then it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x^*_0, x^*_1, j, \tau, i), i)$ to obtain $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$. Otherwise, it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x^j_{i,0}, x^j_{i,1}, j, \tau^*_i, i), i)$ to obtain the ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$, which is sent to the adversary.

6. For every function query f, the challenger first executes $\mathsf{FE.KeyGen}(\mathsf{FE.msk}, f)$ to obtain $\mathsf{FE}.sk_f$. It then sends the functional key $\mathsf{MIFE}_{c+1}.sk_f = \mathsf{FE}.sk_f$ to the adversary.

7. Adversary outputs b'.

Lemma 12. *For any PPT adversary $\mathcal{A}$, we have $|\mathsf{Adv}^{2^+.3}_{\mathcal{A}} - \mathsf{Adv}^{2^+.2}_{\mathcal{A}}| \leq \epsilon$ assuming the ϵ-selective security of* FE.

The proof of the above lemma is similar to the proof of Lemma 6.
$\mathsf{Hybrid}_{2^+.4}$ for $\overrightarrow{j} \in [q]^{c+1}$:

1. Adversary submits q number of message queries to the challenger. Denote $(x^j_{i,0}, x^j_{i,1})$ to be the j^{th} message query corresponding to the i^{th} position, where $j \in [q], i \in [c+1]$.

2. Challenger executes $\mathsf{MIFE}_c.\mathsf{Setup}(1^\lambda)$ to obtain $\mathsf{MIFE}_c.\mathsf{msk}$. It then executes $\mathsf{FE.Setup}(1^\lambda)$ to obtain $(\mathsf{FE.pk}, \mathsf{FE.msk})$. Finally, the challenger sets $\mathsf{MIFE}_{c+1}.\mathsf{msk}$ to be $(\mathsf{MIFE}_c.\mathsf{msk}, \mathsf{FE.pk}, \mathsf{FE.msk})$.

3. It sets $\overrightarrow{j} = (j_1, \ldots, j_{c+1})$, with $j_i = q-1$ for all $i \in [c+1]$. It initializes $\mathcal{I}^*$ to be $(j_1 - 1, \ldots, j_{c+1} - 1)$.

4. It picks $\tau^*_2, \ldots, \tau^*_{c+1}$ at random with $\tau^*_i \in \{0,1\}^\lambda$.

5. The challenger then generates the challenge ciphertexts as follows. For every $i \in [c+1], j \in [q]$,

   (a) $i = 1$ and $j \neq j_1$:
   - It draws the PRF key $K \in \{0,1\}^\lambda$ at random.
   - It then executes $\mathsf{MIFE}_c.\mathsf{KeyGen}\big(\mathsf{MIFE}_c.\mathsf{msk}, G\big)$ to obtain $\mathsf{MIFE}_c.sk_G$, where $G = \mathsf{HybGenCT}^{(c,1)}_{[x^j_{1,0}, x^j_{1,1}, j, K, \mathsf{FE.pk}, \mathcal{I}^*]}$ is defined in Figure 4.
   - It sends the challenge ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^* = \mathsf{MIFE}_c.sk_G$ to the adversary.

   (b) $i = 1$ and $j = j_1$:
   - It draws the PRF key $K \in \{0,1\}^\lambda$ at random.
   - Compute ct^* to be $\mathsf{FE.Enc}(\mathsf{FE.pk}, (x^{j_1}_{1,1}, \ldots, x^{j_{c+1}}_{c+1,1}); \mathsf{PRF}(K, \tau^*_2 || \cdots || \tau^*_{c+1}))$.
   - Puncture the PRF key K at the point $(\tau^*_2 || \cdots || \tau^*_{c+1})$ to obtain K^*.
   - Execute $\mathsf{MIFE}_c.\mathsf{KeyGen}(\mathsf{MIFE}_c.\mathsf{msk}, G)$ to obtain $\mathsf{MIFE}_c.sk_G$, where $G = \mathsf{HybGenCT}^{(c,3)}_{[x^j_{1,0}, x^j_{1,1}, j, K^*, \mathsf{FE.pk}^*, \mathcal{I}^*, \mathsf{ct}^*]}$ is defined in Figure 6.
   - It sends the challenge ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^* = \mathsf{MIFE}_c.sk_G$ to the adversary.

   (c) $2 \leq i \leq c+1$ and $j \neq j_i$:
   - It picks τ at random.
   - If $x^j_{i,0}$ is of the form $(x^*_0, x^*_1, j, \tau, i-1)$ then it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x^*_0, x^*_1, j, \tau, i), i)$ to obtain $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$. Otherwise, it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x^j_{i,0}, x^j_{i,1}, j, \tau, i), i)$ to obtain the ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$, which is sent to the adversary.

   (d) $2 \leq i \leq c+1$ and $j = j_i$:

- If $x^j_{i,0}$ is of the form $(x^*_0, x^*_1, j, \tau, i-1)$ then it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x^*_0, x^*_1, j, \tau, i), i)$ to obtain $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$. Otherwise, it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x^j_{i,0}, x^j_{i,1}, j, \tau^*_i, i), i)$ to obtain the ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$, which is sent to the adversary.

6. For every function query f, the challenger first executes $\mathsf{FE.KeyGen}(\mathsf{FE.msk}, f)$ to obtain $\mathsf{FE}.sk_f$. It then sends the functional key $\mathsf{MIFE}_{c+1}.sk_f = \mathsf{FE}.sk_f$ to the adversary.
7. Adversary outputs b'.

Lemma 13. *For any PPT adversary $\mathcal{A}$, we have $|\mathsf{Adv}^{2^+.5}_{\mathcal{A}} - \mathsf{Adv}^{2^+.4}_{\mathcal{A}}| \leq \epsilon$ assuming the ϵ-selective security of F.*

The proof of the above lemma is similar to the proof of Lemma 4.
$\mathsf{Hybrid}_{2^+.5}$ for $\overrightarrow{j} \in [q]^{c+1}$:

1. Adversary submits q number of message queries to the challenger. Denote $(x^j_{i,0}, x^j_{i,1})$ to be the j^{th} message query corresponding to the i^{th} position, where $j \in [q], i \in [c+1]$.
2. Challenger executes $\mathsf{MIFE}_c.\mathsf{Setup}(1^\lambda)$ to obtain $\mathsf{MIFE}_c.\mathsf{msk}$. It then executes $\mathsf{FE.Setup}(1^\lambda)$ to obtain $(\mathsf{FE.pk}, \mathsf{FE.msk})$. Finally, the challenger sets $\mathsf{MIFE}_{c+1}.\mathsf{msk}$ to be $(\mathsf{MIFE}_c.\mathsf{msk}, \mathsf{FE.pk}, \mathsf{FE.msk})$.
3. It sets $\overrightarrow{j} = (j_1, \ldots, j_{c+1})$, with $j_i = q-1$ for all $i \in [c+1]$. It initializes $\mathcal{I}^*$ to be $(j_1 - 1, \ldots, j_{c+1} - 1)$.
4. The challenger then generates the challenge ciphertexts as follows. For every $i \in [c+1], j \in [q]$,
   (a) $i = 1$:
   - It draws the PRF key $K \in \{0,1\}^\lambda$ at random.
   - It then executes $\mathsf{MIFE}_c.\mathsf{KeyGen}\big(\mathsf{MIFE}_c.\mathsf{msk}, G\big)$ to obtain $\mathsf{MIFE}_c.sk_G$, where $G = \mathsf{HybGenCT}^{(c,4)}_{[x^j_{1,0}, x^j_{1,1}, j, K, \mathsf{FE.pk}, \mathcal{I}^*]}$ is defined in Figure 7.
   - It sends the challenge ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^* = \mathsf{MIFE}_c.sk_G$ to the adversary.

   (b) $2 \leq i \leq c+1$:
   - It picks τ at random.
   - If $x^j_{i,0}$ is of the form $(x^*_0, x^*_1, j, \tau, i-1)$ then it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x^*_0, x^*_1, j, \tau, i), i)$ to obtain $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$. Otherwise, it executes $\mathsf{MIFE}_c.\mathsf{Enc}(\mathsf{MIFE}_c.\mathsf{msk}, (x^j_{i,0}, x^j_{i,1}, j, \tau, i), i)$ to obtain the ciphertext $\mathsf{MIFE}_{c+1}.\mathsf{ct}^*$, which is sent to the adversary.
5. For every function query f, the challenger first executes $\mathsf{FE.KeyGen}(\mathsf{FE.msk}, f)$ to obtain $\mathsf{FE}.sk_f$. It then sends the functional key $\mathsf{MIFE}_{c+1}.sk_f = \mathsf{FE}.sk_f$ to the adversary.
6. Adversary outputs b'.

Lemma 14. *For any PPT adversary $\mathcal{A}$, we have $|\mathsf{Adv}^{2^+.5}_{\mathcal{A}} - \mathsf{Adv}^{3}_{\mathcal{A}}| \leq \epsilon$ assuming the ϵ-security of F.*

$\underline{\mathsf{HybGenCT}^{(c,4)}_{[x_1,x'_1,i_1,K^*,\mathsf{FE.pk},\mathcal{I}]}}$

Input: $\Big((x_2, x'_2, j_2, \tau_2, i_2), \ldots, (x_{c+1}, x'_{c+1}, j_{c+1}, \tau_{c+1}, i_{c+1})\Big)$.

- $\forall k \in \{2, \ldots, c+1\}$, if $i_k \neq k$, then output $\perp$
- $\mathcal{I}' = (j_1 - 1, \ldots, j_n - 1)$
- If $\mathsf{LessThan}(\mathcal{I}', \mathcal{I}, c+1, q) = 1$ or $\mathcal{I} = (q-1, \ldots, q-1)$ then
  - $\mathsf{FE.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.pk}, (x'_1, \ldots, x'_{c+1}); \mathsf{PRF}(K^*, \tau_2 || \cdots || \tau_{c+1}))$
- Else,
  - $\mathsf{FE.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.pk}, (x_1, \ldots, x_{c+1}); \mathsf{PRF}(K^*, \tau_2 || \cdots || \tau_{c+1}))$
- Output $\mathsf{FE.ct}$.

Figure 7

6 Multi-Input FE from Single-Input FE

In Section 4, we gave a general transformation from a secret-key MiFE scheme for c-ary functions to another secret-key MiFE scheme for $c+1$-ary functions. Using this transformation, we now give a construction of a secret-key MiFE scheme for functions with $n = \mathsf{poly}(\lambda)$ arity. Later, in Section 7, we will use this construction to obtain our main result on $i\mathrm{O}$. We will also consider different instantiations of this construction which yield new results on constant-ary MiFE from standard assumptions.

More concretely, starting from a single-ary FE scheme, we apply our transformation repeatedly, in an iterative manner, to obtain an n-ary MiFE scheme. Recall that our transformation consists of two steps: the first step, described in 3.1, adds function hiding property to the underlying c-ary MiFE scheme. The second step, described in Section 4, amplifies the arity by 1 by "knitting" the function-hiding c-ary MiFE scheme with a single-ary public key FE scheme. By relying on the correctness and the security of each of these steps, we can prove the correctness and security of the overall construction.

However, there is a possible cause for concern: efficiency. We need to argue that every iteration does not add a multiplicative overhead to the size of the parameters in the MiFE scheme. Indeed, if the converse were to happen, then we would only be able to achieve a constant-ary MiFE scheme starting from a single-ary FE scheme. As we show later, by carefully choosing the instantiation of the public-key FE scheme we can avoid the overhead in every iteration.

We now give more details.

6.1 (Iterated) Construction of MIFE_n

We now construct a n-ary (q,q)-secure MiFE scheme MIFE_n with the message space $\mathcal{X}^n = \{\mathcal{X}^n_\lambda\}_{\lambda\in\mathbb{N}}$ and function space $\mathcal{F}^n = \{\mathcal{F}^n_\lambda\}_{\lambda\in\mathbb{N}}$. For simplicity, we consider $\mathcal{X}_\lambda = \{0,1\}^\lambda$. To obtain this

construction we start with a q-secure[11] public-key FE scheme. This implies a single-ary (q,q)-secure secret-key MiFE scheme, MIFE_1 with $\mathcal{X}^1$ and $\mathcal{F}^1$ to be, respectively, the associated input and the function spaces. We discuss below what $\mathcal{X}^1$ and $\mathcal{F}^1$ should correspond to.

We then repeat the following two steps for $c = 1, \ldots, n$:

1. **Function privacy transformation:** Using the function-privacy transformation presented in Section 3.1, convert the (q,q)-secure MiFE scheme MIFE_c, obtained in the previous iteration, into a function-private (q,q)-secure MiFE scheme $\mathsf{MIFE}_c^{\mathsf{fp}}$, also supporting c-arity functions. The associated function spaces and the message spaces of $\mathsf{MIFE}_c^{\mathsf{fp}}$, denoted by $\mathcal{F}^{\mathsf{fp},c}$ and $\mathcal{X}^{\mathsf{fp},c}$, are defined below in the next step.

2. **Arity amplification:** The function-private c-ary (q,q)-secure MiFE scheme $\mathsf{MIFE}_c^{\mathsf{fp}}$ obtained in the previous step is then transformed into a $c+1$-ary (q,q)-secure MiFE scheme for function space $\mathcal{F}^{c+1}$ and message space $\mathcal{X}^{c+1}$, using the transformation presented in Section 4. In this step, we additionally use a q-secure public-key FE scheme FE. The instantiation of FE in the c^{th} step would essentially be the same as the function space of MIFE_{c+1} and the message space is $\mathcal{X}^{\mathsf{FE},c} = \left\{\left(\mathcal{X}_\lambda^c\right)^c\right\}_{\lambda\in\mathbb{N}}$

   We now define the input message and the function spaces.
   Input message spaces: It follows from the description of the constructions in Sections 3.1 and 4, that:
   $$\mathcal{X}^c = \mathcal{X}^{\mathsf{fp},c} = \mathcal{X}^{c+1} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}},$$
   where $\mathcal{X}_\lambda = \mathcal{X}'_\lambda \cup \mathcal{X}''_\lambda \cup \mathcal{X}'''_\lambda$, and
   - $\mathcal{X}'_\lambda = \{0,1\}^\lambda$,
   - $\mathcal{X}''_\lambda$ contains strings of the form (x_0, x_1, j, τ, i), where $x_0, x_1 \in \mathcal{X}'_\lambda$, $j, \tau \in \{0,1\}^\lambda$ and $i \in \{0,1\}^{\log(n)}$,
   - $\mathcal{X}'''_\lambda$ contains strings of the form $(X_0, X_1, \mathsf{Sym}.K, \mathsf{Sym}.K')$, where $X_0, X_1 \in \mathcal{X}''_\lambda$, $\mathsf{Sym}.K, \mathsf{Sym}.K' \in \{0,1\}^\lambda$.

   Function spaces: Again, it follows from the description of the constructions in Sections 3.1 and 4, that:
   $$\mathcal{F}^{\mathsf{fp},c} = \left\{\mathsf{GenCT}^{(c)}, \mathsf{HybGenCT}^{(c,1)}, \mathsf{HybGenCT}^{(c,2)}, \mathsf{HybGenCT}^{(c,3)}, \mathsf{HybGenCT}^{(c,4)}\right\},$$
   where $\mathsf{GenCT}^{(c)}$, $\mathsf{HybGenCT}^{(c,1)}$, $\mathsf{HybGenCT}^{(c,2)}$, $\mathsf{HybGenCT}^{(c,3)}$ and $\mathsf{HybGenCT}^{(c,4)}$ are described in Figures 3, 4, 5, 6 and 7.
   $$\mathcal{F}^c = \left\{U\right\},$$
   where U is as described in Figure 2. (For simplicity of exposition, we omit the constants for each of the above functions from their respective notations.)

As discussed earlier, the efficiency properties of the underlying public-key FE scheme determines the value of n that we can achieve in the above construction. Consequently, we consider two different instantiations of the underlying public-key FE scheme that yield different results. We discuss these instantiations in the following two subsections.

[11] Throughout this section, we only consider selectively secure public-key FE and secret key MiFE schemes. For simplicity of notation, we omit the use of the word "selective" in the rest of this section and assume that it is implicit.

6.2 MiFE for Poly-arity Functions from Compact FE

We start by stating our main result for secret-key MiFE for polynomial-arity functions.

Theorem 4. *For all $n = \text{poly}(\lambda)$, the proposed scheme MIFE_n in Section 6.1 is (q, q)-secure for any polynomial q, assuming that FE is $\left(1, \frac{1}{(64q)^{(n+1)^2} \cdot 2^\lambda}\right)$-selectively secure compact public-key FE scheme.*

In order to prove the above theorem, we start with the following observation: Gorbunov et al. [GVW12] show how to transform a (1)-secure public-key FE scheme into a (q)-secure public key FE scheme for any polynomial q. This transformation preserves the compactness of the underlying encryption scheme. See Appendix C for more details on this transformation.

Given the above observation, we only need to prove the following theorem:

Theorem 5. *For all $n = \text{poly}(\lambda)$, the proposed scheme MIFE_n in Section 6.1 is (q, q)-secure assuming that FE is $\left(q, \frac{1}{(64q)^{(n+1)^2} \cdot 2^\lambda}\right)$-selectively secure compact public-key FE scheme.*

Proof. The proof of Theorem 5 consists of three steps. First, the correctness of MIFE_n follows from the correctness of the function privacy transformation (Section 3.1) and the correctness of the arity amplification transformation (Section 4). Next, we argue that all the algorithms in MIFE_n are probabilistic polynomial time. And finally, we will discuss the security of MIFE_n.

Efficiency Analysis. We start by establishing some notation.

1. $\mathsf{MaxSize}$: We define the notation $\mathsf{MaxSize}$ for a function family $\mathcal{G}$ to be the maximum size of all the circuits, implementing the functions in $\mathcal{G}$. That is,

$$\mathsf{MaxSize}(\mathcal{G}) = \max_{C \text{ implements } g;\ g \in \mathcal{G}} \Big\{ \mathsf{size}(g) \Big\},$$

   where $\mathsf{size}(g)$ denotes the size of the polynomial sized circuit representing g assuming g is indeed efficiently computable.

2. $\mathsf{RunTime}$: We define the notation $\mathsf{RunTime}$ to denote the size of a circuit corresponding to a particular input length. Suppose A is a circuit and it receives inputs of lengths $z_1, \ldots, z_m$. Then, $\mathsf{RunTime}(A, z_1, \ldots, z_m)$ denotes the size of the circuit A on input length $\sum_{i=1}^{m} z_i$.

We now analyze the run times of the algorithms of (q, q)-secure scheme, MIFE_n. To do this, we need to first estimate the run times of the algorithms[12] of MIFE_c, for $1 \leq c \leq n-1$. We do this recursively. First, the size of the parameters in MIFE_{c+1} is expressed as the size of the parameters in MIFE_c – this is done by looking at the function privacy transformation and the arity amplification steps. Then, we recursively apply this to obtain the run times in MIFE_n expressed in terms of the run times in the q-secure compact public-key functional encryption scheme FE.

We denote the non-function hiding c-ary MiFE in step c as $\mathsf{MIFE}_c = (\mathsf{MIFE}_c.\mathsf{Setup}, \mathsf{MIFE}_c.\mathsf{KeyGen}, \mathsf{MIFE}_c.\mathsf{Enc}, \mathsf{MIFE}_c.\mathsf{Dec})$ for function space $\mathcal{F}^c$ and message space $\mathcal{X}^c$. Further, the function hiding c-ary MiFE is denoted as $\mathsf{MIFE}_c^{\mathsf{fp}} = (\mathsf{MIFE}_c^{\mathsf{fp}}.\mathsf{Setup}, \mathsf{MIFE}_c^{\mathsf{fp}}.\mathsf{KeyGen}, \mathsf{MIFE}_c^{\mathsf{fp}}.\mathsf{Enc}, \mathsf{MIFE}_c^{\mathsf{fp}}.\mathsf{Dec})$ for function space $\mathcal{F}^{\mathsf{fp},c}$ and message space $\mathcal{X}^{\mathsf{fp},c}$.

[12] Here, we abuse the notation and use the terminology 'runtime' (in place of 'size') even though we are referring to circuits.

Before we perform the analysis of the running times of the setup and the encryption algorithms, we first determine the complexity of the function spaces we obtain at every step. This quantity has an effect on the run times of the encryption as well as the key generation algorithms.

Since the message spaces $\mathcal{X}^c$, $\mathcal{X}^{\mathsf{fp},c}$, for $1 \leq c < n$, are all equal to each other, we use ℓ_x to denote the maximum length of all the strings in $\mathcal{X}^c = \mathcal{X}^{\mathsf{fp},c}$. For $1 \leq c \leq n$, we have:

$$\mathsf{MaxSize}(\mathcal{F}^{\mathsf{fp},c}) = \mathsf{RunTime}\big(\mathsf{FE.Enc}, \lambda, c, q, (\ell_x)^{c+1}\big) + p_1(\lambda),$$

where p_1 is a polynomial. Further, we have

$$\mathsf{MaxSize}(\mathcal{F}^{c}) = p_2(\lambda, c, \mathsf{MaxSize}(\mathcal{F}^{\mathsf{fp},c})) + p_3(\lambda),$$

where p_2 and p_3 are polynomials. Combining the above quantities, we get $\mathsf{MaxSize}(\mathcal{F}^c)$ to be a polynomial in (λ, c, q), for every $1 \leq c \leq n$.

1. Run time of Setup algorithm $\mathsf{MIFE}_n.\mathsf{Setup}$: We first focus our attention on the run time of the setup algorithm $\mathsf{MIFE}_{c+1}.\mathsf{Setup}$. We express this in terms of $\mathsf{MIFE}_c.\mathsf{Setup}$. This is done by first expressing $\mathsf{MIFE}_{c+1}.\mathsf{Setup}$ in terms of $\mathsf{MIFE}_c^{\mathsf{fp}}.\mathsf{Setup}$ and then we express $\mathsf{MIFE}_c^{\mathsf{fp}}.\mathsf{Setup}$ in terms of $\mathsf{MIFE}_c.\mathsf{Setup}$.

For $n > c \geq 1$, we have:

1. *Arity amplification step*: The setup algorithm of a the $c+1$-ary scheme internally executes the setup algorithm of the function-private c-ary scheme and the setup algorithm of the compact public-key FE scheme. Thus the run time of this step is

   $$\mathsf{RunTime}(\mathsf{MIFE}_{c+1}.\mathsf{Setup}, \lambda, c{+}1, q) = \mathsf{RunTime}(\mathsf{MIFE}_c^{\mathsf{fp}}.\mathsf{Setup}, \lambda, c, q){+}\mathsf{RunTime}(\mathsf{FE.Setup}, \lambda, q){+}p_4(\lambda)$$

2. *Function privacy transformation step*: The setup algorithm of the function-private c-ary MiFE scheme runs the setup algorithm of (non-function private) setup. The run time of this step is $\mathsf{RunTime}(\mathsf{MIFE}_c^{\mathsf{fp}}.\mathsf{Setup}, \lambda, c, q) = \mathsf{RunTime}(\mathsf{MIFE}_c.\mathsf{Setup}, \lambda, c, q) + p_5(\lambda)$,

where p_4, p_5 are polynomials. Here, $p_4(\lambda)$ subsumes the running time of the setup algorithm of the compact public-key FE scheme and $p_5(\lambda)$ subsumes the running time of the setup algorithm of the symmetric encryption scheme.

If we recursively apply both the above equations, we get $\mathsf{RunTime}(\mathsf{MIFE}_n.\mathsf{Setup}, n, \lambda, q)$ to be just a polynomial in λ, since n is also a polynomial in λ.

2. Run time of Key Generation algorithm $\mathsf{MIFE}_n.\mathsf{KeyGen}$: The run time of the key generation procedure in step $c+1$ depends mainly on the complexity of the function classes in step c. We break this analysis into two steps, by looking first at the arity amplification step and then looking at the function privacy transformation step.

For $n - 1 > c \geq 1$; we have the following:

1. *Arity amplification step*: The key generation algorithm $\mathsf{MIFE}_{c+1}.\mathsf{KeyGen}$ internally invokes the key generation algorithm $\mathsf{MIFE}_c^{\mathsf{fp}}.\mathsf{KeyGen}$. The complexity of this step is hence, $\mathsf{RunTime}($ $\mathsf{FE.KeyGen}, \lambda, c, q, \mathsf{MaxSize}(\mathcal{F}^{\mathsf{fp},c}))$, where F is defined below.

2. *Function privacy transformation step*: The key generation algorithm $\mathsf{MIFE}_c^{\mathsf{fp}}.\mathsf{KeyGen}$ internally invokes the key generation algorithm $\mathsf{MIFE}_c.\mathsf{KeyGen}$. So, the run time is $\mathsf{RunTime}(\mathsf{MIFE}_c.\mathsf{KeyGen}$, $\lambda, c, q, |G|) \leq \mathsf{RunTime}(\mathsf{MIFE}_c.\mathsf{KeyGen}, \lambda, c, q, \mathsf{MaxSize}(\mathcal{F}^c))$,

where $F \in \mathcal{F}^{c+1}$ and $G \in \mathcal{F}^{\mathsf{fp},c}$. Further, we denote ℓ_f to be $\max_{g \in \mathcal{F}^n}\{\mathsf{size}(g)\}$ From this, we can show the following.

$$\mathsf{RunTime}(\mathsf{MIFE}_n.\mathsf{KeyGen}, \lambda, n, q, |f|) \leq \max_{1 \leq c \leq n} \Big\{ \mathsf{RunTime}(\mathsf{FE}.\mathsf{KeyGen}, \lambda, c, q, \mathsf{MaxSize}(\mathcal{F}^{\mathsf{fp},c}), \\ \mathsf{RunTime}(\mathsf{FE}.\mathsf{KeyGen}, \lambda, c, q, \mathsf{MaxSize}(\mathcal{F}^{c})) \Big\}$$

where $f \in \mathcal{F}^n$. Since, $\mathsf{MaxSize}(\mathcal{F}^c)$, for all c, is already a polynomial in the security parameter λ, we have the run time of $\mathsf{MIFE}_{c+1}.\mathsf{KeyGen}$ to be at most a polynomial in (λ, c, q, ℓ_f).

3. Run time of Encryption algorithm $\mathsf{MIFE}_n.\mathsf{Enc}$: The analysis of the run time of encryption algorithm can be done by first expressing the run time of the encryption algorithm $\mathsf{MIFE}_{c+1}.\mathsf{Enc}$, for $1 \leq c < n$, in terms of the run time of key generation algorithm $\mathsf{MIFE}_c^{\mathsf{fp}}.\mathsf{KeyGen}$ and encryption algorithm $\mathsf{MIFE}_c^{\mathsf{fp}}.\mathsf{Enc}$. We then express the run time of $\mathsf{MIFE}_c^{\mathsf{fp}}.\mathsf{Enc}$ in terms of the run time of (non-function private) encryption $\mathsf{MIFE}_c.\mathsf{Enc}$. The analysis of these two steps can be further broken down into two steps, depending on the position with respect to which we encrypt the messages.

1. *Arity amplification step*: We analyze the run time of the algorithm $\mathsf{MIFE}_{c+1}.\mathsf{Enc}$. Consider the following two cases:

   - $i = 1$: In this case, a functional key of $\mathsf{MIFE}_c^{\mathsf{fp}}$ is generated. Thus, the complexity of encrypting the message in this case would essentially be the run time of the key generation algorithm, $\mathsf{MIFE}_c^{\mathsf{fp}}.\mathsf{KeyGen}$ on input functions in the space $\mathcal{F}^{\mathsf{fp},c}$. Thus run time of this step is $\mathsf{RunTime}(\mathsf{MIFE}_c^{\mathsf{fp}}.\mathsf{KeyGen}, \lambda, c, q, \mathsf{MaxSize}(\mathcal{F}^c))$.
   - $i > 1$: In this case, the encryption is computed by invoking the encryption of c-ary MiFE scheme. Thus, the run time of this step is $\mathsf{RunTime}(\mathsf{MIFE}_c^{\mathsf{fp}}.\mathsf{Enc}, \lambda, c, q, \ell_x)$.

   Thus, total run time of this step is

   $$\mathsf{RunTime}(\mathsf{MIFE}_c^{\mathsf{fp}}.\mathsf{KeyGen}, \lambda, c, q, \mathsf{MaxSize}(\mathcal{F}^c)) + \mathsf{RunTime}(\mathsf{MIFE}_c^{\mathsf{fp}}.\mathsf{Enc}, \lambda, c, q, \ell_x) + p_6(\lambda),$$

   where p_6 is a polynomial.

2. *Function privacy transformation step*: We analyze the run time of $\mathsf{MIFE}_c^{\mathsf{fp}}.\mathsf{Enc}$. Note that in this step, the encryption w.r.t position 1 and any other position is done by invoking the encryption $\mathsf{MIFE}_c.\mathsf{Enc}$. The only difference is between the messages that are encrypted w.r.t position 1 and those encrypted w.r.t any other position. In both the cases, the length of the messages is at most ℓ_x. Thus, the run time of this step is $\mathsf{RunTime}(\mathsf{MIFE}_c^{\mathsf{fp}}.\mathsf{Enc}, c, \lambda, q, \ell_x) + p_7(\lambda)$, where p_7 is a polynomial.

We are now ready to express the run time of $\mathsf{MIFE}_{c+1}.\mathsf{Enc}$ in terms of $\mathsf{MIFE}_c.\mathsf{Enc}$. Moreover, we have calculated the run time of $\mathsf{MIFE}_c^{\mathsf{fp}}.\mathsf{KeyGen}$ to be a polynomial in (λ, c), call it p_8. We have for $m \in \mathcal{X}^{c+1}$,

$$\mathsf{RunTime}(\mathsf{MIFE}_{c+1}.\mathsf{Enc}, \lambda, c, q, |m|) \leq \mathsf{RunTime}(\mathsf{MIFE}_c.\mathsf{Enc}, \lambda, c, q, \ell_x) + p_6(\lambda) + p_7(\lambda) + p_8(\lambda)$$

Hence, we have $\mathsf{RunTime}(\mathsf{MIFE}_n.\mathsf{Enc}, \lambda, q, c, |m|)$, where $m \in \mathcal{X}^n$, to be a polynomial in $\mathsf{RunTime}($ $\mathsf{FE}.\mathsf{Enc}, \lambda, q, c, \ell_x)$, n, and λ which is a polynomial in (λ, c, q, ℓ_x).

4. Run time of Decryption algorithm $\mathsf{MIFE}_n.\mathsf{Dec}$: To analyze the run time of $\mathsf{MIFE}_n.\mathsf{Dec}$, we consider the following two steps. For $n \geq c > 1$, we have

1. *Arity amplification step*: The decryption algorithm, $\mathsf{MIFE}_{c+1}.\mathsf{Dec}$ executes the decryption $\mathsf{MIFE}_c^{\mathsf{fp}}.\mathsf{Dec}$ and FE decryption $\mathsf{FE}.\mathsf{Dec}$. Thus, the run time of this step is at most $\mathsf{RunTime}(\mathsf{MIFE}_c^{\mathsf{fp}}.\mathsf{Dec}, t_1, t_2)$, where $t_1 = \mathsf{RunTime}(\mathsf{MIFE}_c^{\mathsf{fp}}.\mathsf{Enc}, \lambda, c, q, \ell_x)$ and $t_2 = \mathsf{RunTime}(\mathsf{MIFE}_c^{\mathsf{fp}}.\mathsf{KeyGen}, \lambda, c, q, \mathsf{MaxSize}(\mathcal{F}^{\mathsf{fp},c}))$.

2. *Function privacy transformation step*: The decryption algorithm, $\mathsf{MIFE}_c^{\mathsf{fp}}.\mathsf{Dec}$ internally executes $\mathsf{MIFE}_c.\mathsf{Dec}$ and so, the run time of this step is at most $\mathsf{RunTime}(\mathsf{MIFE}_c.\mathsf{Dec}, \mathsf{RunTime}(\mathsf{MIFE}_c.\mathsf{Enc}, \lambda, c, q, \ell_x), \mathsf{RunTime}(\mathsf{MIFE}_c.\mathsf{KeyGen}, \lambda, c, q, \mathcal{F}^c))$

This is nothing but a polynomial in $(\lambda, c, q, \ell_x, \ell_f)$ since the size of the ciphertexts and the functional keys are, respectively, polynomials in (λ, c, q, ℓ_x) and (λ, c, q, ℓ_f).

The above four bullets prove that all the four algorithms of MIFE_n run in probabilistic polynomial time.

Security Analysis. We argue the security of MIFE_n by invoking the security of the arity amplification and the function privacy steps. We set $\delta_1 = \epsilon$, where FE is (q, ϵ)-secure. From Theorem 2, we have that for $1 \leq c < n$, the scheme $\mathsf{MIFE}_c^{\mathsf{fp}}$ yielded by the c^{th} function privacy step in the construction of Section 6.1 is $(q, q, \delta_c^{\mathsf{fp}})$-secure, where $\delta_c^{\mathsf{fp}} = \frac{\delta_c}{4}$. From Theorem 3, we have that for $1 \leq c < n$, the scheme MIFE_{c+1} yielded by the c^{th} arity amplification step is (q, q, δ_{c+1})-secure, where $\delta_{c+1} = (8 + 5q^{c+1})\delta_c^{\mathsf{fp}}$-secure. Thus, we have,

$$\begin{aligned}
\delta_n &= (8 + 5q^n)\delta_{n-1}^{\mathsf{fp}} \\
&= (8 + 5q^n)4\delta_{n-1} \\
&\leq (64q)^n \delta_{n-1} \\
&\leq \prod_{c=1}^{n-1} (64q)^{c+1} \delta_1 \\
&\leq (64q)^{2+\ldots+n} \frac{1}{(64q)^{(n+1)^2} 2^\lambda} \\
&\leq \mathsf{negl}(\lambda)
\end{aligned}$$

where negl is a negligible function. This proves that MIFE_n is (q, q)-secure.

From the efficiency analysis argument presented above we have that the algorithms in the scheme MIFE_n run in probabilistic polynomial time. Further, from the security analysis, we have that MIFE_n is (q, q)-secure. This completes the proof of Theorem 5. □

6.2.1 MiFE for Constant-arity Functions from Standard Assumptions

In the previous subsection, we showed that applying the iterated construction, presented in Section 6.1, on a compact FE scheme yields a n-ary MiFE scheme, where n is a polynomial in λ. We now turn to applying the same construction but on a non compact FE scheme. In this case the iterated construction yields a n-ary MiFE scheme, where n is a constant. The main reason why we only get constant arity is because the size of the parameters grow at an exponential rate.

Formally, we prove the following theorem.

Theorem 6. *For any constant n, the proposed scheme MIFE_n in Section 6.1 is (q,q)-selectively secure assuming that FE is a q-selectively secure (not necessarily compact) public-key FE scheme.*

Combining Theorem 6 with [SS10, GVW12], we obtain the following result.

Corollary 1. *For any polynomial $q = q(\lambda)$, there exists a (q,q)-selectively secure secret-key MiFE scheme for constant-arity functions, assuming the existence of semantically-secure public-key encryption.*

Further, combining Theorem 6 with the public-key FE scheme of [GGH+13b], we obtain the following result.

Corollary 2. *Assuming the existence of indistinguishability obfuscation and one-way functions, there exists an unbounded selectively-secure secret-key MiFE scheme for constant-arity functions.*

We now give the proof of Theorem 6.

Proof. As before, the proof of the theorem consists of three steps. The first step, which is the correctness of MIFE_n follows from the correctness of the function privacy transformation (Section 3.1) and the correctness of the arity amplification transformation (Section 4). The next two steps are, respectively, the efficiency and security analysis of MIFE_n.

Efficiency Analysis. As before, we first perform the analysis of the complexity of function space at each step. Once we do this, we can refer back to the analysis of setup, key generation, encryption and decryption as done when we instantiated with the compact scheme since the exact same analysis will hold even in this case. We borrow the notation of $\mathsf{MaxSize}$ and $\mathsf{RunTime}$ as defined in Section 6.2. Further, as before, we use ℓ_x to denote the maximum length of all the strings in $\mathcal{X}^c = \mathcal{X}^{\mathsf{fp},c}$, for every $1 \leq c \leq n$.

For $1 \leq c \leq n$, we have:

$$\mathsf{MaxSize}(\mathcal{F}^{\mathsf{fp},c}) = \mathsf{RunTime}\big(\mathsf{FE.Enc}, \lambda, q, \mathsf{MaxSize}(\mathcal{F}^{c+1}), (\ell_x)^{c+1}\big) + p_1(\lambda),$$

where p_1 is a polynomial. Note that in the above expression, unlike the compact FE case, $\mathsf{MaxSize}(\mathcal{F}^{\mathsf{fp},c})$ has dependence on $\mathsf{MaxSize}(\mathcal{F}^{c+1})$. Further, we have

$$\mathsf{MaxSize}(\mathcal{F}^{c}) = p_2(\lambda, \mathsf{MaxSize}(\mathcal{F}^{\mathsf{fp},c})) + p_3(\lambda),$$

where p_2 and p_3 are polynomials. Combining the above quantities, we get

$$\mathsf{MaxSize}(\mathcal{F}^{c}) = p_9(\lambda, \mathsf{MaxSize}(\mathcal{F}^{c+1}), c, \ell_x),$$

where p_9 is a polynomial. We have $\mathsf{MaxSize}(\mathcal{F}^1)$ to be exponential in n, where the iterated construction yields a n-ary MiFE scheme. When n is a constant, $\mathsf{MaxSize}(\mathcal{F}^1)$ is polynomial in (λ, c, ℓ_x).

Security Analysis. The security argument in this case is exactly the same as the security argument presented in the proof of Theorem 5. That is, we can show that the construction in Section 6.1 yields a (q,q,δ_n)-secure scheme, where δ_n is at most $(64q)^{n^2}\epsilon$ and ϵ is a negligible function in λ such that FE is (q,ϵ)-selectively secure. Since n is a constant, we have that δ_n is negligible in λ.

From the above arguments, we have that the algorithms in MIFE_n run in probabilistic polynomial time and further, MIFE_n is q-secure. This completes the proof of Theorem 6 □

7 Indistinguishability Obfuscation from Compact FE

In this Section, we establish our main result on $i\mathrm{O}$ from compact public-key FE. To obtain this result, we rely on following theorem of Goldwasser et al. [GGG+14]:

Theorem 7 ([GGG+14]). *Assuming the existence of a $(1,2)$-secure secret-key MiFE scheme for general functions with arity $(n+1)$, there exists an indistinguishability obfuscator for all functions with input length n.*

Very briefly, their construction can be described as follows: suppose we wish to obfuscate any circuit with input length n. Then, start with a $n+1$-ary secret-key MiFE scheme. The obfuscation of a circuit C consists of $(\mathsf{MIFE}_{n+1}.sk_U, \mathsf{MIFE}_{n+1}.\mathsf{ct}_{1,C}, \mathsf{MIFE}_{n+1}.\mathsf{ct}_{1,0}, \mathsf{MIFE}_{n+1}.\mathsf{ct}_{1,1}, \ldots, \mathsf{MIFE}_{n+1}.\mathsf{ct}_{n,0}, \mathsf{MIFE}_{n+1}.\mathsf{ct}_{n,1})$, where $\mathsf{MIFE}_2.sk_U$ is the functional key of the universal function that takes as input $(C, x_1, \ldots, x_n)$ and outputs $C(x)$ with the binary representation of x being $x_1||\cdots||x_n$, and $\mathsf{MIFE}_{n+1}.\mathsf{ct}_{i,b}$ is the encryption of b corresponding to the $(i-1)^{th}$ input of C. To evaluate the obfuscation on an input x, the evaluator first considers the bit representation of x, denote this by $x_1, \ldots, x_n$. It then executes $\mathsf{MIFE}_{n+1}.\mathsf{Dec}(\mathsf{MIFE}_{n+1}.\mathsf{msk}, \mathsf{MIFE}_{n+1}.\mathsf{ct}_{1,f}, \mathsf{MIFE}_{n+1}.\mathsf{ct}_{2,x_1}, \ldots, \mathsf{MIFE}_{n+1}.\mathsf{ct}_{n+1,x_n})$ and the result is $C(x)$.

$i\mathrm{O}$ from compact public-key FE. Combining Theorem 7 with Theorem 4 for the case of $q=2$, we obtain an indistinguishability obfuscation scheme, denoted by $i\mathrm{O}$, for $P/poly$. Formally, we state the theorem below.

Theorem 8. *Assuming the $(2, \frac{1}{(128)^{n^2}2^\lambda})$-security of compact public-key selectively secure FE public key FE scheme for polynomial time computable functions, the scheme $i\mathrm{O}$ is an indistinguishability obfuscation scheme for $P/poly$.*

8 Compact FE from RE for Turing Machines

In this Section, we construct a compact bounded-query public-key FE scheme for randomized encodings [IK00, AIK06] for turing machines. Our construction proceeds in two steps: first, we show how to obtain a compact bounded-query FE for NC^1 functionalities. In the next step, we apply the general bootstrapping theorem to convert FE for NC^1 to general functonalities [GHRW14, ABSV14]. This step preserves the compactness of the underlying FE.

We start with the definition of randomized encoding for TMs that we need for our construction. We then describe the first step of our construction.

8.1 Randomized Encoding for TMs

A randomized encoding (RE) of a function is a simplified representation of the computation of the function on a particular input such that (i) computing this representation must be much simpler than actually carrying out the computation and (ii) the only information revealed by the representation is the output of computation and nothing else. Traditionally, circuit implementation of the functions, over which RE is computed, was considered. Recently, the works of [BGL+15, CHJV15, KLW15] considered the scenario where the functions are implemented by Turing machines. We define this notion, termed as RE for TMs, below. We associate the Turing machine with this RE to be $\mathcal{M}$.

1. **Encoding.** $\mathsf{RE}_{TM}.\mathsf{Encode}(1^\lambda, M, x, T)$: On input security parameter λ, Turing machine $M \in \mathcal{M}$, input $x \in \{0,1\}^*$, and time bounded T output the encoding $\widetilde{M(x)}$.
2. **Decoding.** $\mathsf{RE}_{TM}.\mathsf{Decode}(\widetilde{M(x)})$: On input encoding $\widetilde{M(x)}$, output the decoded value y.

There are three important properties that any RE for TMs need to satisfy.

Correctness. This property states that the output of $\mathsf{RE}_{TM}.\mathsf{Decode}(\widetilde{M(x)})$ is $M(x)$ if the machine M on input x runs for at most T steps, where $\widetilde{M(x)} \leftarrow \mathsf{RE}_{TM}.\mathsf{Encode}(1^\lambda, M, x, T)$ and $M \in \mathcal{M}$.

Efficiency. This property intuitively says that the run time of the encode procedure depends on the size of the Turing machine and the input length and in particular, is independent of the computation time of the TM. Formally, the run time of $\mathsf{RE}_{TM}.\mathsf{Encode}$ on input $(1^\lambda, M \in \mathcal{M}, x, T)$ is a polynomial in $(\lambda, |M|, |x|, \log(T))$. Furthermore, the run time of $\mathsf{RE}_{TM}.\mathsf{Decode}$ on input $(\widetilde{M(x)})$ is a polynomial in $(\lambda, |M|, |x|, t)$, where $\widetilde{M(x)} \leftarrow \mathsf{RE}_{TM}.\mathsf{Encode}(1^\lambda, M \in \mathcal{M}, x, T)$ and $t \leq T$ is the time taken by M to execute on x.

Indistinguishability security. The security requirement states that given a machine $M \in \mathcal{M}$ and two inputs x_0 and x_1, the corresponding encodings $\widetilde{M(x_0)}$ is computationally indistinguishable from the encoding $\widetilde{M(x_1)}$. Of course this itself is insufficient since the adversary could look at the sizes of the encodings, outputs of the decode procedure or even its run time to distinguish the two encodings. Hence, we place the additional constraint that $|x_0| = |x_1|$ and further $M(x_0) = M(x_1)$, with the run time of M on x_0 being the same as the run time of M on x_1. Formally, for any sufficiently large security parameter $\lambda \in \mathbb{N}$, PPT adversary $\mathcal{A}$, the following holds for any negligible function negl.

$$\mathsf{Pr}\Big[b' = b : (M, x_0, x_1, T, \mathsf{st}) \leftarrow \mathcal{A}(1^\lambda);\ b \xleftarrow{\$} \{0,1\};\ \widetilde{M(x_b)} \leftarrow \mathsf{RE}_{TM}.\mathsf{Encode}(1^\lambda, M, x_b, T);\ b' \leftarrow \mathcal{A}(\mathsf{st}, \widetilde{M(x_b)})\Big] \leq \mathsf{negl}(\lambda)$$

This completes the description of RE for TMs.

8.2 Construction

Our construction makes use of RE for TMs as well as a q-query *semi-compact* (see Definition 4) public-key FE scheme for NC^1 functions. We note that a semi-compact single-key FE scheme was constructed by Goldwasser et al. [GKP+13]. Using the result of Gorbunov [GVW12], we can amplify the number of key queries to any bounded polynomial. We note that the GVW transformation preserves the semi-compactness property of Goldwasser et al. See Appendix C for more details.

We now proceed to describe our construction.

Notation. The randomized encoding for TMs scheme is denoted by $\mathsf{RE}_{TM} = (\mathsf{RE}_{TM}.\mathsf{Encode}, \mathsf{RE}_{TM}.\mathsf{Decode})$. We denote the semi-compact FE scheme to be $\mathsf{SCFE} = (\mathsf{SCFE.Setup}, \mathsf{SCFE.KeyGen}, \mathsf{SCFE.Enc}, \mathsf{SCFE.Dec})$ for NC^1 functionalities. The compact scheme we construct is denoted by $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$. The function space is denoted by $\mathcal{F}$. We denote by ℓ_f by the maximum length of the function in $\mathcal{F}$ (as a function of security parameter) and we denote out_f to be the output length of all the functions in $\mathcal{F}$.

The formal description of the algorithms are given below.

$\mathsf{FE.Setup}(1^\lambda)$: On input security parameter λ, first sample a PRF key $K \in \{0,1\}^\lambda$ at random. Then execute $\mathsf{RE}_{TM}.\mathsf{Encode}(1^\lambda, \mathsf{SetupFunc}, K)$, where $\mathsf{SetupFunc}$ is as described in Figure 8, to obtain $\widetilde{\mathsf{SetupFunc}}$ Then execute $\mathsf{SCFE.Setup}(1^\lambda; \mathsf{PRF}(K,i))$, to obtain $(\mathsf{SCFE.mpk}_i, \mathsf{SCFE.msk}_i)$, for $i \in \mathsf{out}_f$. Output $(\mathsf{mpk} = \widetilde{\mathsf{SetupFunc}}, \mathsf{FE.msk} = (\mathsf{SCFE.msk}_1, \ldots, \mathsf{SCFE.msk}_{\mathsf{out}_f}))$.

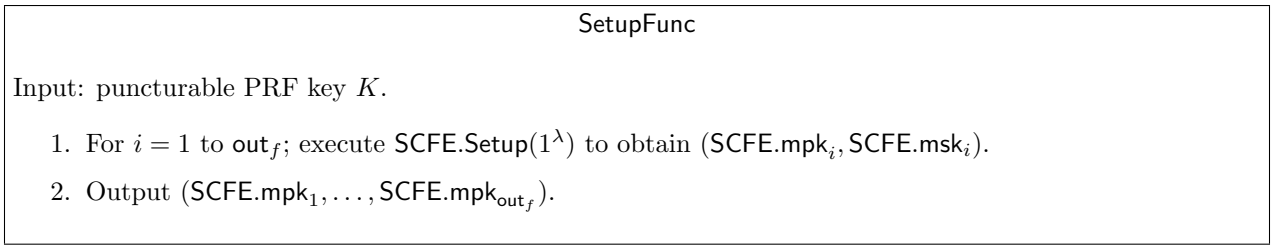

Figure 8

$\mathsf{FE.KeyGen}(\mathsf{FE.msk}, f)$: On input $\mathsf{FE.msk}$ and a function f, parse $\mathsf{FE.msk}$ as $(\mathsf{SCFE.msk}_1, \ldots, \mathsf{SCFE.msk}_{\ell_f})$. For every $i \in \mathsf{out}_f$, where out_f denotes the output length of f, execute $\mathsf{SCFE.KeyGen}(\mathsf{SCFE.msk}_i, G_{f,i})$ to obtain $\mathsf{SCFE}.sk_{G,i}$, where $G_{f,i}$ takes as input m and outputs the i^{th} bit of $f(m)$. Output $\mathsf{FE}.sk_f = (\mathsf{SCFE}.sk_{G,1}, \ldots, \mathsf{SCFE}.sk_{G,\mathsf{out}_f})$.

$\mathsf{FE.Enc}(\mathsf{mpk}, m)$: On input public key mpk and message m, first sample a puncturable PRF key $K \in \{0,1\}^\lambda$. Then, executes the encode algorithm $\mathsf{RE}_{TM}.\mathsf{Encode}(1^\lambda, \mathsf{EncFunc}, (\mathsf{mpk}, K, (m, \perp)))$, where $\mathsf{EncFunc}$ is as described in Figure 10, to obtain $\widetilde{\mathsf{EncFunc}}$. Output $\mathsf{FE.ct} = \widetilde{\mathsf{EncFunc}}$.

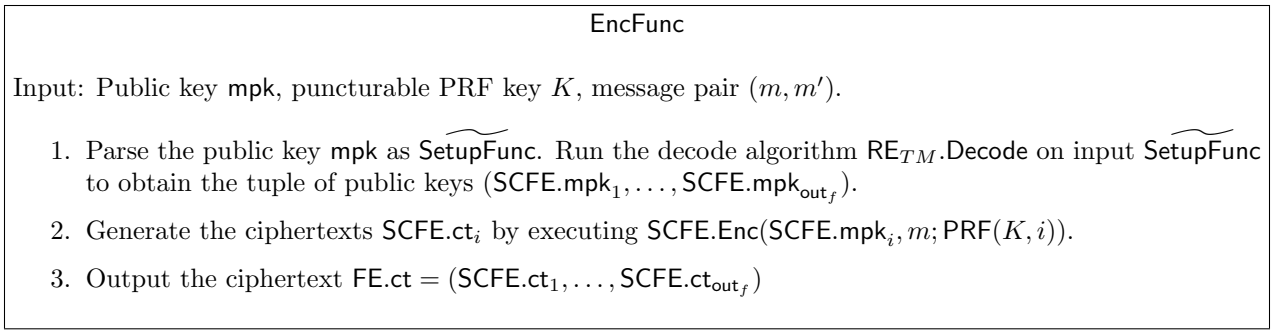

Figure 9

$\mathsf{FE.Dec}(\mathsf{FE}.sk_f, \mathsf{FE.ct})$: On input functional key $\mathsf{FE}.sk_f$ and ciphertext $\mathsf{FE.ct}$, first parse $\mathsf{FE}.sk_f$ as $(\mathsf{SCFE}.sk_{G,1}, \ldots, \mathsf{SCFE}.sk_{G,\mathsf{out}_f})$. Then parse $\mathsf{FE.ct}$ as $\widetilde{\mathsf{EncFunc}}$. First, execute the decode algorithm $\mathsf{RE}_{TM}.\mathsf{Decode}$ on input $(\widetilde{\mathsf{EncFunc}})$ to obtain $\mathsf{FE.ct} = (\mathsf{SCFE.ct}_1, \ldots, \mathsf{SCFE.ct}_{\mathsf{out}_f})$. Then execute the

decryption algorithm $\mathsf{SCFE.Dec}$ on input $(\mathsf{SCFE}.sk_{G,i}, \mathsf{SCFE.ct}_i)$, for $i \in [\mathsf{out}_f]$, to obtain b_i. Output $b_1 \cdots b_{\mathsf{out}_f}$.

This completes the description of the scheme.

Correctness. The correctness of the above scheme is easy to verify.

Efficiency. We need to argue that the encryption algorithm runs in time polynomial in $(\lambda, |m|)$, where m is the message to be encrypted. Here, the polynomial should be chosen even before the function space is designed. To analyze this quantity, we first look at the public key output by the setup algorithm. The public key is an encoding of the TM, represented in Figure 8. The size of this TM is a polynomial in $(\lambda, \log(\mathsf{out}_f))$, which is just another polynomial in λ. And hence, the size of the encoding is also a polynomial in λ. Thus, the size of the public key mpk is a polynomial in λ. We now analyze the running time of the encryption algorithm. It is a polynomial in $(\lambda, |\mathsf{mpk}|, |\mathsf{EncFunc}|, |m|)$, where $\mathsf{EncFunc}$ is represented in Figure 10. We have argued that $|\mathsf{mpk}|$ just depends on λ. We further observe that $|\mathsf{EncFunc}|$ is a polynomial in $(\lambda, |m|, \log(\mathsf{out}_f))$, which shows that the complexity of the encryption algorithm is a polynomial $(\lambda, |m|)$.

Theorem 9. *If RE_{TM} is a secure RE for TMs and SCFE is a semi-compact (q)-secure public-key FE, then FE is a compact (q)-secure public key FE scheme.*

We prove Theorem 9 in Appendix B.

Acknowledgements

The authors thank Amit Sahai for useful discussions. We also thank Mark Zhandry for sharing his insightful comments on the existence of order-revealing encryption.

References

[AAB+13] Shashank Agrawal, Shweta Agrawal, Saikrishna Badrinarayanan, Abishek Kumarasubramanian, Manoj Prabhakaran, and Amit Sahai. Function private functional encryption and property preserving encryption: New definitions and positive results. *IACR Cryptology ePrint Archive*, 2013:744, 2013.

[AB15] Benny Applebaum and Zvika Brakerski. Obfuscating circuits via composite-order graded encoding. In *TCC*, 2015.

[ABG+13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. *IACR Cryptology ePrint Archive*, 2013:689, 2013.

[ABSV14] Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. The trojan method in functional encryption: From selective to adaptive security, generically. *IACR Cryptology ePrint Archive*, 2014:917, 2014.

[AGIS14] Prabhanjan Ananth, Divya Gupta, Yuval Ishai, and Amit Sahai. Optimizing obfuscation: Avoiding barrington's theorem. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 646–658. ACM, 2014.

[AGVW13] Shweta Agrawal, Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption: New perspectives and lower bounds. In Canetti and Garay [CG13], pages 500–518.

[AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. *Computational Complexity*, 15(2):115–162, 2006.

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In Yehuda Lindell, editor, *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, volume 8349 of *Lecture Notes in Computer Science*, pages 52–73. Springer, 2014.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2001.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In *Public-Key Cryptography–PKC 2014*, pages 501–519. Springer, 2014.

[BGK+14] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 221–238, 2014.

[BGL+15] Nir Bitansky, Sanjam Garg, Huijia Lin, Rafael Pass, and Siddartha Telang. Succinct randomized encodings and their applications. In *STOC*, 2015.

[BKS15] Zvika Brakerski, Ilan Komargodski, and Gil Segev. From single-input to multi-input functional encryption in the private-key setting. *IACR Cryptology ePrint Archive*, 2015:158, 2015.

[BLR+15] Dan Boneh, Kevin Lewi, Mariana Raykova, Amit Sahai, Mark Zhandry, and Joe Zimmerman. Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation. In *EUROCRYPT*, 2015.

[BR14] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In *TCC*, pages 1–25, 2014.

[BRS13] Dan Boneh, Ananth Raghunathan, and Gil Segev. Function-private identity-based encryption: Hiding the function in functional encryption. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 461–478, 2013.

[BS15] Zvika Brakerski and Gil Segev. Function-private functional encryption in the private-key setting. In *TCC*, 2015.

[BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. *IACR Cryptology ePrint Archive*, 2015:163, 2015.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *Advances in Cryptology-ASIACRYPT 2013*, pages 280–300. Springer, 2013.

[BWZ14] Dan Boneh, David J. Wu, and Joe Zimmerman. Immunizing multilinear maps against zeroizing attacks. *IACR Cryptology ePrint Archive*, 2014:930, 2014.

[CG13] Ran Canetti and Juan A. Garay, editors. *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, volume 8043 of *Lecture Notes in Computer Science*. Springer, 2013.

[CHJV15] Ran Canetti, Justin Holmgren, Abhishek Jain, and Vinod Vaikuntanathan. Indistinguishability obfuscation of iterated circuits and RAM programs. In *STOC*, 2015.

[CHL+15] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In *EUROCRYPT*, 2015.

[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 Canetti and Garay [CG13], pages 519–535.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 476–493, 2013.

[CLT14] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Cryptanalysis of two candidate fixes of multilinear maps over the integers. *IACR Cryptology ePrint Archive*, 2014:975, 2014.

[CLTV15] Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. In *TCC*, 2015.

[GGG+14] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 578–602. Springer, 2014.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2013.

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

[GGHR14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, pages 74–94, 2014.

[GGHZ14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mark Zhandry. Fully secure functional encryption without obfuscation. *IACR Cryptology ePrint Archive*, 2014:666, 2014.

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

[GHL+14] Craig Gentry, Shai Halevi, Steve Lu, Rafail Ostrovsky, Mariana Raykova, and Daniel Wichs. Garbled RAM revisited. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 405–422, 2014.

[GHMS14] Craig Gentry, Shai Halevi, Hemanta K. Maji, and Amit Sahai. Zeroizing without zeroes: Cryptanalyzing multilinear maps without encodings of zero. *IACR Cryptology ePrint Archive*, 2014:929, 2014.

[GHRW14] Craig Gentry, Shai Halevi, Mariana Raykova, and Daniel Wichs. Outsourcing private RAM computation. *IACR Cryptology ePrint Archive*, 2014:148, 2014.

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

[GLOS15] Sanjam Garg, Steve Lu, Rafail Ostrovsky, and Alessandra Scafuro. Garbled RAM from one-way functions. In *STOC*, 2015.

[GLSW14] Craig Gentry, Allison B. Lewko, Amit Sahai, and Brent Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. *IACR Cryptology ePrint Archive*, 2014:309, 2014.

[GLW14] Craig Gentry, Allison Lewko, and Brent Waters. Witness encryption from instance independent assumptions. In *Advances in Cryptology–CRYPTO 2014*, pages 426–443. Springer, 2014.

[Gol09] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*, volume 2. Cambridge university press, 2009.

[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.

[IK00] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 294–304, 2000.

[KLW15] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for turing machines with unbounded memory. In *STOC*, 2015.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 669–684. ACM, 2013.

[KSY15] Ilan Komargodski, Gil Segev, and Eylon Yogev. Functional encryption for randomized functionalities in the private-key setting from minimal assumptions. In *TCC*, 2015.

[LO13] Steve Lu and Rafail Ostrovsky. How to garble RAM programs. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 719–734, 2013.

[PST14] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 500–517, 2014.

[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.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 475–484. ACM, 2014.

[SZ14] Amit Sahai and Mark Zhandry. Obfuscating low-rank matrix branching programs. Technical report, Cryptology ePrint Archive, Report 2014/773, 2014. http://eprint. iacr. org, 2014.

[Wat14] Brent Waters. A punctured programming approach to adaptively secure functional encryption. Cryptology ePrint Archive, Report 2014/588, 2014.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167, 1986.

[Zim15] Joe Zimmerman. How to obfuscate programs directly. In *EUROCRYPT*, 2015.

A MiFE Function Privacy Transformation: Proof of Theorem 2

The first hybrid, Hybrid_1, corresponds to the real experiment when the challenger uses the challenge bit 0 to encrypt the messages and generate the functional keys. The last hybrid, Hybrid_5, corresponds to the real experiment when the challenger uses the challenge bit 1 to encrypt the messages and generate the functional keys. We then prove the indistinguishability of Hybrid_i and Hybrid_{i+1}. This further proves that the hybrids Hybrid_1 and Hybrid_5 are computationally indistinguishable which completes the proof.

$\underline{\mathsf{Hybrid}_1}$: This corresponds to the function privacy game of the MiFE scheme, where the challenger uses bit 0 to encrypt the messages and generate the functional keys.

The challenger first executes $\mathsf{FP.Setup}(1^\lambda)$ to obtain $\mathsf{FP.msk}$. It also obtains $\mathsf{Sym}.K \leftarrow \mathsf{Sym.Setup}(1^\lambda)$ and $\mathsf{Sym}.K' \leftarrow \mathsf{Sym.Setup}(1^\lambda)$. Upon receiving a challenge message query of the form $(m_0^{(i)}, m_1^{(i)})$, for all $i \in [c]$, the challenger does the following. It encrypts the message $m_{1,0}$ by executing $\mathsf{FP.Enc}(\mathsf{FP.msk}, (m_{1,0}, \perp, \mathsf{Sym}.K, \perp), 1)$ and encrypts $m_{i,0}$, for $i \in [c]$ and $i \neq 1$, by executing $\mathsf{FP.Enc}(\mathsf{FP.msk}, m_{i,0}, i)$. Further, the challenger on receiving function pair queries of the form (f_0, f_1), generates the functional key of f_0. The output of the adversary is the output of the hybrid.

$\underline{\mathsf{Hybrid}_2}$: In this hybrid, the challenger computes the symmetric encryption ciphertexts $\mathsf{Sym.ct}, \mathsf{Sym.ct}'$ in the functional key to be the encryptions of f_0 and f_1 respectively, where (f_0, f_1) is the function query submitted by the adversary. More formally, the hybrid is described as follows.

Upon receiving a function pair query (f_0, f_1), the challenger first generates the master secret key by executing $\mathsf{FP.Setup}(1^\lambda)$ to obtain $\mathsf{FP.msk}$. $\mathsf{Sym}.K \leftarrow \mathsf{Sym.Setup}(1^\lambda)$ and $\mathsf{Sym}.K' \leftarrow \mathsf{Sym.Setup}(1^\lambda)$. It also obtains $\mathsf{Sym.ct} \leftarrow \mathsf{Sym.Enc}(\mathsf{Sym}.K, f_0)$ and $\mathsf{Sym.ct}' \leftarrow \mathsf{Sym.Enc}(\mathsf{Sym}.K, f_1)$. Finally, the challenger generates the functional key $\mathsf{FP}.sk_f^*$, by executing $\mathsf{FP.KeyGen}(\mathsf{FP.msk}, U_{[\mathsf{Sym.ct},\mathsf{Sym.ct}']})$, where $U_{[\mathsf{Sym.ct},\mathsf{Sym.ct}']}$ is as defined in Figure 2. The rest of the hybrid (for instance, handling message queries) are as in the previous hybrid.

The computational indistinguishability of Hybrid_1 and Hybrid_2 follows from the security of symmetric encryption scheme.

$\underline{\mathsf{Hybrid}_3}$: In this hybrid, the challenge ciphertexts are computed as follows. Upon receiving a challenge message query of the form $\big((m_{1,0}, m_{1,1}), \ldots, (m_{c,0}, m_{c,1})\big)$, the challenger generates the ciphertexts $\mathsf{FP.ct}_i^*$ to be the output of $\mathsf{FP.Enc}(\mathsf{FP.msk}, (\perp, m_{1,1}, \perp, \mathsf{Sym}.K'), 1)$, if $i = 1$, or to be the output of $\mathsf{FP.Enc}(\mathsf{FP.msk}, m_{i,1}, i)$ if $i \neq 1$. Here, $\mathsf{FP.msk}$, $\mathsf{Sym}.K$ and $\mathsf{Sym}.K'$ are generated as in the previous hybrid. The rest of the hybrid (for instance, handling function queries) are as in Hybrid_2.

The computational indistinguishability of Hybrid_2 and Hybrid_3 follows from the adaptive (resp., selective security) of the c-ary MiFE scheme.

$\underline{\mathsf{Hybrid}_4}$: In this hybrid, the challenger generates the functional keys as follows. Upon receiving a function pair query (f_0, f_1), the challenger first generates the master secret key by executing $\mathsf{FP.Setup}(1^\lambda)$ to obtain $\mathsf{FP.msk}$. $\mathsf{Sym}.K \leftarrow \mathsf{Sym.Setup}(1^\lambda)$ and $\mathsf{Sym}.K' \leftarrow \mathsf{Sym.Setup}(1^\lambda)$. It also obtains $\mathsf{Sym.ct} \leftarrow \mathsf{Sym.Enc}(\mathsf{Sym}.K, f_1)$ and $\mathsf{Sym.ct}' \leftarrow \mathsf{Sym.Enc}(\mathsf{Sym}.K, f_1)$. Finally, the challenger generates the functional key $\mathsf{FP}.sk_f^*$, by executing $\mathsf{FP.KeyGen}(\mathsf{FP.msk}, U_{[\mathsf{Sym.ct},\mathsf{Sym.ct}']})$, where

$U_{[\mathsf{Sym.ct},\mathsf{Sym.ct}']}$ is as defined in Figure 2. The rest of the hybrid (for instance, handling message queries) are as in the previous hybrid.

Note that the only difference between Hybrid_2 and Hybrid_4 is that in Hybrid_4, both $\mathsf{Sym.ct}$ and $\mathsf{Sym.ct}'$ are generated to be the encryptions of the function f_1, whereas in Hybrid_2, $\mathsf{Sym.ct}$ is generated to be the encryption of f_0 and $\mathsf{Sym.ct}'$ is generated to be the encryption of f_1.

The computational indistinguishability of Hybrid_3 and Hybrid_4 follows from the security of the symmetric encryption scheme.

$\underline{\mathsf{Hybrid}_5}$: This hybrid corresponds to the real function privacy game, where the challenger uses bit 1 to generate the challenge ciphertexts and the functional keys.

The challenger first executes $\mathsf{FP.Setup}(1^\lambda)$ to obtain $\mathsf{FP.msk}$. It also obtains $\mathsf{Sym}.K \leftarrow \mathsf{Sym.Setup}(1^\lambda)$ and $\mathsf{Sym}.K' \leftarrow \mathsf{Sym.Setup}(1^\lambda)$. Upon receiving a challenge message query of the form $(m_0^{(i)}, m_1^{(i)})$, for all $i \in [c]$, the challenger does the following. It encrypts the message $m_1^{(1)}$ by executing $\mathsf{FP.Enc}(\mathsf{FP.msk}, (m_{1,1}, \bot, \mathsf{Sym}.K, \bot), 1)$ and encrypts $m^{(i)}$, for $i \in [c]$ and $i \neq 1$, by executing $\mathsf{FP.Enc}(\mathsf{FP.msk}, m_1^{(i)}, i)$. The rest of the hybrid is as in Hybrid_4.

The computational indistinguishability of Hybrid_4 and Hybrid_5 follows from the selective security of the c-ary MiFE scheme.

Further note that if the scheme NFP is $(q_{\mathsf{key}}, q_{\mathsf{msg}}, \epsilon)$-secure then the resulting FP is $(q_{\mathsf{key}}, q_{\mathsf{msg}}, \frac{\epsilon}{4})$-secure.

B Proof of Theorem 9

We first describe the hybrids and then informally sketch the proof of the indistinguishability of hybrids. The first hybrid corresponds to the real experiment where the challenge bit is 0. The last hybrid corresponds to the real experiment where the challenge bit is 1. To argue indistinguishability, we use the security of the RE for TMs primitive, (semi-succinct) FE scheme, and the puncturable PRF scheme.

We formally present the hybrids below.

$\underline{\mathsf{Hybrid}_1}$: This corresponds to the real experiment where the challenge bit 0 is used by the challenger. That is, upon receiving a message pair query (m_0, m_1) from the adversary, the challenger will compose the challenge ciphertext by encrypting the message m_0.

$\underline{\mathsf{Hybrid}_2}$: In this hybrid, the challenger upon receiving the message pair query (m_0, m_1) from the adversary, composes the challenge ciphertext by executing the encode algorithm $\mathsf{RE}_{TM}.\mathsf{Encode}(1^\lambda, \mathsf{EncFunc}, (\mathsf{mpk}, K, (m_0, \underline{m_1})))$, where $\mathsf{EncFunc}$ is as described in Figure 10, to obtain $\widetilde{\mathsf{EncFunc}}$.The challenger then sends the challenge ciphertext $\mathsf{FE.ct}^* = \widetilde{\mathsf{EncFunc}}$ to the adversary. The rest of the hybrid is the same as Hybrid_1.

The indistinguishability of Hybrid_1 and Hybrid_2 follows from the security of randomized encodings. This follows from the observation that the output $\mathsf{EncFunc}(\mathsf{mpk}, (m_0, \bot))$ is the same as the output $\mathsf{EncFunc}(\mathsf{mpk}, (m_0, m_1))$, because the program $\mathsf{EncFunc}$ essentially ignores the message m_1.

Before describe Hybrid_3, we describe a set of intermediate hybrids, for all $j \in [\mathsf{out}_f + 1]$.

$\underline{\mathsf{Hybrid}_{2.j.1}}$: In this hybrid, the challenger upon receiving the message pair query (m_0, m_1), composes the challenge ciphertext as follows. It executes the encode algorithm $\mathsf{RE}_{TM}.\mathsf{Encode}(1^\lambda, \mathsf{EncFunc}[j], (\mathsf{mpk}, K, (m_0, m_1)))$, where $\mathsf{EncFunc}$ is as described in Figure 12, to obtain $\widetilde{\mathsf{EncFunc}}[j]$.

The hybrids Hybrid_2 and $\mathsf{Hybrid}_{2.1.1}$ are computationally indistinguishable assuming the security of randomized encodings. This again follows from the observation that the output $\mathsf{EncFunc}(\mathsf{mpk}, (m_0, m_1))$ is the same as the output $\mathsf{EncFunc}[1](\mathsf{mpk}, (m_0, m_1))$.

$\mathsf{EncFunc}[j]$

Input: Public key mpk, puncturable PRF key K, message pair (m, m').

1. Parse the public key mpk as $\widetilde{\mathsf{SetupFunc}}$. Run the decode algorithm $\mathsf{RE}_{TM}.\mathsf{Decode}$ on input $\widetilde{\mathsf{SetupFunc}}$ to obtain the tuple of public keys $(\mathsf{SCFE.mpk}_1, \ldots, \mathsf{SCFE.mpk}_{\mathsf{out}_f})$.
2. For $i = 1$ to out_f, do the following:
   (a) If $i < j$, generate the ciphertext $\mathsf{SCFE.ct}_i$ by executing $\mathsf{SCFE.Enc}(\mathsf{SCFE.mpk}_i, m'; \mathsf{PRF}(K, i))$.
   (b) If $i \geq j$, generate the ciphertext $\mathsf{SCFE.ct}_i$ by executing $\mathsf{SCFE.Enc}(\mathsf{SCFE.mpk}_i, m_0; \mathsf{PRF}(K, i))$.
3. Output the ciphertext $\mathsf{FE.ct} = (\mathsf{SCFE.ct}_1, \ldots, \mathsf{SCFE.ct}_{\mathsf{out}_f})$

Figure 10

$\underline{\mathsf{Hybrid}_{2.j.2}}$: In this hybrid, the challenger upon receiving the message pair query (m_0, m_1) does the following. It picks two puncturable keys $K_1, K_2 \in \{0,1\}^\lambda$. It then punctures the key at the point j. It then samples the key $\mathsf{SCFE.Setup}(1^\lambda; \mathsf{PRF}(K_1, j))$ to obtain $\mathsf{SCFE.mpk}_j$. It further generates the ciphertext $\mathsf{SCFE.ct}_j$ by executing $\mathsf{SCFE.Enc}(\mathsf{SCFE.mpk}_j, m_0; \mathsf{PRF}(K_2, j))$. It then punctures both the keys K_1 and K_2 at the point j to respectively obtain the PRF keys K_1^* and K_2^*. It then constructs the functions $\mathsf{SetupFunc}^*[j, \mathsf{SCFE.mpk}_j]$ and $\mathsf{EncFunc}^*[j, \mathsf{SCFE.ct}_j]$ as described in Figure 11 and Figure 12 respectively. The challenger then generates the public key by executing $\mathsf{RE}_{TM}.\mathsf{Encode}(1^\lambda, \mathsf{SetupFunc}^*[j, \mathsf{SCFE.mpk}_j], K_1^*)$ to obtain mpk and then it generates the ciphertext by executing the encode algorithm $\mathsf{RE}_{TM}.\mathsf{Encode}(1^\lambda, \mathsf{EncFunc}^*[j, \mathsf{SCFE.ct}_j], (\mathsf{mpk}, K_2^*, (m_0, m_1)))$ to obtain $\mathsf{FE.ct}^*$. The challenger then sends both mpk and $\mathsf{FE.ct}^*$ across to the adversary. The rest of the hybrid is the same as before.

To argue indistinguishability of $\mathsf{Hybrid}_{2.j.1}$ and $\mathsf{Hybrid}_{2.j.2}$, we need to introduce one more intermediate hybrid $\mathsf{Hybrid}^{0.5}_{2.j.1}$ which is the same as $\mathsf{Hybrid}_{2.j.2}$ except that the puncturing of the PRF key is only done for the setup algorithm and the PRF key for the encryption algorithm remains intact. The indistinguishability of $\mathsf{Hybrid}_{2.j.1}$ and $\mathsf{Hybrid}^{0.5}_{2.j.1}$, and similarly, that of $\mathsf{Hybrid}^{0.5}_{2.j.1}$ and $\mathsf{Hybrid}_{2.j.2}$, follows from the security of randomized encodings for Turing machines.

$\underline{\mathsf{Hybrid}_{2.j.3}}$: This hybrid is the same as in the previous hybrid, except that the public key $\mathsf{SCFE.mpk}_j$ and the challenge ciphertext $\mathsf{FE.ct}^*$ are computed by using uniformly chosen randomness and not pseudorandom values as before.

The indistinguishability of $\mathsf{Hybrid}_{2.j.2}$ and $\mathsf{Hybrid}_{2.j.3}$ follows from the security of puncturable

$\mathsf{SetupFunc}[j, \mathsf{SCFE.mpk}_j]^*$

Input: punctured PRF key K^*.

1. For $i = 1$ to out_f; execute $\mathsf{SCFE.Setup}(1^\lambda)$ to obtain $(\mathsf{SCFE.mpk}_i, \mathsf{SCFE.msk}_i)$.
2. Output $(\mathsf{SCFE.mpk}_1, \ldots, \mathsf{SCFE.mpk}_{\mathsf{out}_f})$.

Figure 11

$\mathsf{EncFunc}[j, \mathsf{SCFE.ct}_j]^*$

Input: Public key mpk, puncture PRF key K^*, message pair (m, m').

1. Parse the public key mpk as $\widetilde{\mathsf{SetupFunc}}$. Run the decode algorithm $\mathsf{RE}_{TM}.\mathsf{Decode}$ on input $\widetilde{\mathsf{SetupFunc}}$ to obtain the tuple of public keys $(\mathsf{SCFE.mpk}_1, \ldots, \mathsf{SCFE.mpk}_{\mathsf{out}_f})$.
2. For $i = 1$ to out_f, do the following:
   (a) If $i < j$, generate the ciphertext $\mathsf{SCFE.ct}_i$ by executing $\mathsf{SCFE.Enc}(\mathsf{SCFE.mpk}_i, m'; \mathsf{PRF}(K, i))$.
   (b) If $i > j$, generate the ciphertext $\mathsf{SCFE.ct}_i$ by executing $\mathsf{SCFE.Enc}(\mathsf{SCFE.mpk}_i, m_0; \mathsf{PRF}(K, i))$.
3. Output the ciphertext $\mathsf{FE.ct} = (\mathsf{SCFE.ct}_1, \ldots, \mathsf{SCFE.ct}_{\mathsf{out}_f})$

Figure 12

PRFs. However this cannot be directly argued and like before, we need to consider an intermediate hybrid, where we first compute $\mathsf{SCFE.mpk}_j$ using uniformly chosen randomness but still compute $\mathsf{FE.ct}^*$ using pseudorandom values.

$\mathsf{Hybrid}_{2.j.4}$: This hybrid is the same as in the previous hybrid, upon receiving message query of the form (m_0, m_1), the challenger does the following. It computes the challenge ciphertext $\mathsf{FE.ct}^*$ to be a $\mathsf{FE.Enc}$ encryption of m_1, instead of m_0 as in the previous hybrid.

The indistinguishability of $\mathsf{Hybrid}_{2.j.3}$ and $\mathsf{Hybrid}_{2.j.4}$ follows from the semantic security of the semi-succinct functional encryption scheme. This is because, the adversary is only handed over keys for functions f such that $f(m_0) = f(m_1)$.

$\mathsf{Hybrid}_{2.j.5}$: This hybrid is the same as in the previous hybrid, except that the public key $\mathsf{SCFE.mpk}_j$ and ciphertext $\mathsf{FE.ct}^*$ are computed using pseudorandom values instead of uniformly chosen randomness. That is, upon receiving a message query of the form (m_0, m_1), the challenger does the following: it generates $\mathsf{SCFE.mpk}_j$ to be the output of $\mathsf{SCFE.Setup}(1^\lambda; \mathsf{PRF}(K_1, j))$ and it generates $\mathsf{SCFE.Enc}(\mathsf{SCFE.mpk}_j, m_1; \mathsf{PRF}(K_2, j))$ to obtain $\mathsf{FE.ct}^*$.

The indistinguishability of $\mathsf{Hybrid}_{2.j.4}$ and $\mathsf{Hybrid}_{2.j.5}$ follows from the security of puncturable PRFs. As before, this will require one intermediate hybrid where only $\mathsf{SCFE.mpk}_j$ is switched from being computed using uniformly chosen randomness to pseudorandomness.

$\mathsf{Hybrid}_{2.j.6}$: In this hybrid, the challenger upon receiving a message query (m_0, m_1) does the following. It essentially "unpunctures" the values $\mathsf{SCFE.mpk}_j$ and $\mathsf{SCFE.ct}_j$ respectively from the

programs $\mathsf{SetupFunc}^*[j, \mathsf{SCFE.mpk}_j]$ and $\mathsf{EncFunc}^*[j, \mathsf{SCFE.ct}_j]$. That is, the public key mpk will now be just a execution of $\mathsf{RE}_{TM}.\mathsf{Encode}(1^\lambda, \mathsf{SetupFunc}, K_1)$ and $\mathsf{FE.ct}^*$ will now be an execution of $\mathsf{RE}_{TM}.\mathsf{Encode}(1^\lambda, \mathsf{EncFunc}[j+1], (\mathsf{mpk}, K_2, (m_0, m_1)))$, where K_1 and K_2 are punctured PRF keys. The challenger then sends mpk and $\mathsf{FE.ct}^*$ to the adversary. The rest of the hybrid is the same as before.

The indistinguishability of $\mathsf{Hybrid}_{2.j.5}$ and $\mathsf{Hybrid}_{2.j.6}$, via an intermediate hybrid, can be argued using the security of randomized encodings for Turing machines. Further, the hybrids $\mathsf{Hybrid}_{2.j.6}$ and $\mathsf{Hybrid}_{2.j+1.1}$ are identical to each other.

Hybrid_3: This hybrid corresponds to the real experiment where the challenger uses the bit 1 to compute the challenge ciphertexts.

The indistinguishability of $\mathsf{Hybrid}_{2.\mathsf{out}_f+1.1}$ and Hybrid_3 follows from the security of randomized encodings for Turing machines.

This shows that the output distributions of Hybrid_2 and Hybrid_3 are computationally indistinguishable which proves the security of our construction.

C From Compact (1)-Secure FE to Compact (q_{key})-secure FE

In this section, we show how to transform any compact 1-secure (single key) FE into a compact q_{key}-secure FE, where q_{key} is the number of function queries. Since we only deal with the public key setting in this section, we will not explicitly mention this fact from now on. The transformation is broken down into the following steps.

1. **Boosting security:** We employ the technique of De Caro et al. [CIJ+13] to transform any compact IND-secure single key FE scheme into another compact single key FE scheme that satisfies *simulation security*.[13] Both the schemes are associated with NC^1 functionalities. The resulting simulation secure scheme for NC^1 only satisfies selective security irrespective of the security (selective or adaptive) of the original scheme.

2. **Single to bounded number of keys for NC^1 functionalities:** We then apply the transformation of Gorbunov et al. [GVW12] to achieve compact simulation secure q_{key}-*secure* FE scheme starting from any compact simulation secure *single key* FE scheme. Furthermore, the original scheme and the resulting scheme are only associated with NC^1 functionalities. Since it was not shown by GVW that their transformation is compact preserving, we show this below. We will not go into all the details of the GVW transformation and only mention the essential details. We use the notation of GVW to denote that OneQFE is the single key FE scheme and the q_{key}-secure FE scheme resulting from their transformation to be BDFE.

   Suppose the encryption of BDFE, namely $\mathsf{BDFE.Enc}$, takes as input m. We the express it in terms of $\mathsf{OneQFE.Enc}$, encryption algorithm of OneQFE, as follows. We use the notation of $\mathsf{RunTime}$ which was defined in Section 6.2.

$$\mathsf{RunTime}(\mathsf{BDFE.Enc}, \lambda, q_{\mathsf{key}}, |m|) = N \cdot \mathsf{RunTime}(\mathsf{OneQFE.Enc}, \lambda, |m| \times S),$$

[13] We note that the transformation of De Caro et al. [CIJ+13] is compactness preserving although it is not explicitly mentioned in their work.

where $N = \theta(D^2 q_{\mathsf{key}}^2 t)$, $S = \theta(\lambda q_{\mathsf{key}}^2)$, with D being the degree of the circuits representing the function space of OneQFE and $t = \theta(\lambda q_{\mathsf{key}}^2)$. We note that D is at most the security parameter λ since the circuits are in NC^1.

3. **Bootstraping from NC^1 to P:** The last step involves applying the bootstrapping mechanism, using the compactness-preserving transformation of [GHRW14, ABSV14],[14] to obtain a compact FE scheme for arbitrary functionalities from a compact FE scheme for NC^1. In more detail, we can obtain a compact q_{key}-secure (IND-secure) FE scheme for arbitrary functionalities starting from any compact q_{key}-secure (IND-secure) FE scheme for NC^1. We note that the compact q_{key}-secure FE scheme we obtain in the previous step is already a IND-secure scheme since simulation security implies indistinguishability security.

This completes the description of the transformation.

[14] The fact that this transformation preserves the compactness of the underlying scheme is not explicitly mentioned in these works.