Multi-Input Functional Encryption

Shafi Goldwasser* Vipul Goyal† Abhishek Jain‡ Amit Sahai§

Abstract

We introduce the problem of Multi-Input Functional Encryption, where a secret key SK_f can correspond to an n-ary function f that takes multiple ciphertexts as input. Multi-input functional encryption is a general tool for computing on encrypting data which allows for *mining aggregate information from several different data sources* (rather than just a single source as in single input functional encryption). We show wide applications of this primitive to running SQL queries over encrypted database, non-interactive differentially private data release, delegation of computation, etc.

We formulate both indistinguishability-based and simulation-based definitions of security for this notion, and show close connections with indistinguishability and virtual black-box definitions of obfuscation. Assuming indistinguishability obfuscation for circuits, we present constructions achieving indistinguishability security for a large class of settings. We show how to modify this construction to achieve simulation-based security as well, in those settings where simulation security is possible. Assuming differing-inputs obfuscation [Barak et al., FOCS'01], we also provide a construction with similar security guarantees as above, but where the keys and ciphertexts are *compact*.

*MIT and Weizmann. shafi@csail.mit.edu

†Microsoft Research, India. vipul@microsoft.com

‡Boston University and MIT. abhishek@csail.mit.edu

§UCLA. sahai@cs.ucla.edu

1 Introduction

Traditionally, encryption has been used to secure a communication channel between a unique sender-receiver pair. In recent years, however, our networked world has opened up a large number of new usage scenarios for encryption. For example, a single piece of encrypted data, perhaps stored in an untrusted cloud, may need to be used in different ways by different users. To address this issue, the notion of functional encryption (FE) was developed in a sequence of works [SW05, GPSW06, BW07, KSW08, LOS+10, BSW11, O'N10]. In functional encryption, the owner of the master secret key MSK can create a secret key SK_f for any function f from a family $\mathcal{F}$. Given any ciphertext CT with underlying plaintext x, using SK_f a user can efficiently compute $f(x)$. The security of FE requires that the adversary "does not learn anything" about x, other than the computation result $f(x)$.

How to define "does not learn anything about" x is a fascinating question which has been addressed by a number of papers, with general formal definitions first appearing in [BSW11, O'N10]. The definitions range from requiring a strict *simulation* of the view of the adversary, which enlarges the range of applications, but has been shown to either necessitate a secret key whose size grows with the number of ciphertexts that will ever be released [BSW11, BO13] (or a ciphertext whose size grows with the number of functions for which secret keys will ever be released [AGVW13, CIJ+13]), to an *indistinguishability* of ciphertexts requirement which supports the release of an unbounded number of function keys and ciphertexts.

Functional encryption seems to offer the perfect non-interactive solution to many problems which arise in the context of delegating services to outside servers. A typical example is the delegation of spam filtering to an outside server as follows: Alice publishes her public key online and gives the spam filter a key for the filtering function; users sending email to Alice will encrypt the email with her public key. The spam filter can now determine by itself, for each email, whether to pass it along to Alice's mailbox or to deem it as spam, but without ever learning anything else about Alice's email. This example inherently requires computing a function f on a *single* ciphertext.

Multi-Input Functional Encryption. It is less clear, however, how to define or achieve functional encryption in the context of computing a function defined over *multiple* plaintexts given their corresponding ciphertexts, or further, given their ciphertexts each encrypted under a different key. Yet, these settings, which we formalize as *Multi-Input Functional Encryption* (MI-FE), encompass a vast landscape of applications, going way beyond delegating computation to an untrusted server or cloud. Multi-input functional is a very general tool for computing on encrypting data, which allows for *mining aggregate information from several different data sources* (rather than just a single source as in single input functional encryption).

Let us begin by clarifying the setting of Multi-Input Functional Encryption: Let f be an n-ary function where $n > 1$ can be a polynomial in the security parameter. In MI-FE, the owner of a master secret key MSK can derive special keys SK_f whose knowledge enables the computation of $f(x_1, \ldots, x_n)$ from n ciphertexts $\mathsf{CT}_1, \ldots, \mathsf{CT}_n$ of underlying messages $x_1, \ldots, x_n$ with respect to the same master secret key MSK. We allow the different ciphertexts c_i to be each encrypted under a *different* encryption key EK_i to capture the setting in which each ciphertext was generated by an entirely different party.

Let us illustrate a few settings that illustrate the applicability of MI-FE.

Example 1: Running SQL Queries on Encrypted Database. Suppose we have an encrypted database. A natural goal in this scenario would be to allow a party Alice to perform a certain class of general SQL queries over this database (e.g., Alice may only be authorized to access records created on a certain date). If we use ordinary functional encryption, Alice would need to obtain a separate secret key for every possible valid SQL query, a potentially exponentially large set. Multi-input functional encryption allows us to address this problem in

a flexible way. We highlight two aspects of how MI-FE can apply to this example:

- Let f be the function where $f(Q, x)$ first checks if Q is a valid SQL query from the allowed class, and if so $f(Q, x)$ is the output of the query Q on the database x. Now, if we give the secret key SK_f *and* the encryption key EK_1 to Alice, then Alice can choose a valid query Q and encrypt it under her encryption key EK_1 to obtain ciphertext CT_1. Then she could use her secret key SK_f on ciphertexts CT_1 and CT_2, where CT_2 is the encrypted database, to obtain the results of the SQL query.
- Furthermore, if the database is dynamic (rather than static) with individual entries being added, modified, or, deleted, the most natural way to build such a database would be to have different ciphertexts for each entry in the database. In this case, for a database of size n, we could let f be an $(n+1)$-ary function where $f(Q, x_1, \ldots, x_n)$ is the result of a (valid) SQL query Q on the database $(x_1, \ldots, x_n)$.

Example 2: Computing over Encrypted Data Stream. Suppose ciphertexts correspond to a stream of encrypted phone calls (or video frames produced by surveillance cameras), produced separately by several different devices. Law enforcement agencies may require the ability to run algorithms which check the calls or videos for suspicious activities (these algorithms need to analyze sequences of calls or frames rather than individual calls/frames) in which case (and only in this case) court orders can be obtained to decrypt the phone calls (or videos) in their entirety. Here, the need is to compute a function $f(p_1, \ldots, p_n)$ where p_i is the i'th phone call, encrypted to form the ciphertext c_i .

More generally, suppose ciphertexts $c_1, \ldots, c_n$ correspond to a *list* of encrypted inputs to some algorithm, e.g. a list of edges $x_1, \ldots, x_n$ in a graph for a routing algorithm f. Then, we need to run the algorithm $f(x_1, \ldots, x_n)$ across multiple ciphertexts. It is likely that this type of algorithm would be the rule rather than the exception in the context of algorithms run over large inputs.

Example 3: Non-Interactive Differentially Private Data Release. Suppose there are several hospitals each of which holds a collection of individual blood samples. They would like to participate in clinical trials performed by various researchers. The hospitals cannot simply release the blood samples records because of various patient privacy laws. However, the hospitals are willing to allow a clinical study researcher to compute an aggregate function f over multiple samples x_i to learn $y = f(x_1, \ldots, x_n)$ as long as f achieves a sufficient level of privacy.

While such a scenario is addressed by differential privacy [DMNS06], existing solutions require each hospital to interact with the researcher in every trial (potentially via a multi-party computation protocol when several hospitals are involved). Indeed, it is known that non-cryptographic methods for allowing the hospitals to *non-interactively* prepare their records in a way that would later allow for meaningful and diverse research studies *must* incur high accuracy loss [DNR$^+$09].

Multi-input functional encryption can address this problem by having the hospitals encrypt the samples x_i to obtain ciphertexts CT_i, and publish all the ciphertexts. This step can be performed by the hospitals non-interactively *before* any research trial f is decided (in contrast to the standard differential-privacy setting where f is decided upon first and then the "differentially private" information collection algorithm takes place). Later, a researcher who wishes to compute an algorithm f' (that is guaranteed to provide sufficient privacy) would be given a secret key $\mathsf{SK}_{f'}$ (potentially by a trusted agency such as the government) that she can use to obtain the output of her algorithm on the blood samples. In this manner, we can obtain high accuracy while still guaranteeing good (computational) privacy.

We remark that this example requires MI-FE to support *randomized* functionalities. Our positive results, discussed later, handle this case.

Example 4: Multi-client Delegation of Computation. In a multi-client delegation scheme

[CKKC13], multiple weak clients $C_1, \ldots, C_n$ wish to jointly delegate the computation of an n-ary function f on their inputs $x_1, \ldots, x_n$ to a computationally powerful server. The efficiency requirement of a delegation scheme is that the computation of the clients should be independent of the size of f. From a security viewpoint, we require that a dishonest server should not be able to convince the (honest) clients on an incorrect output.

Multi-input functional encryption provides a natural solution to this problem similar to how single-input functional encryption provides a solution for *single-client* delegation of computation [PRV12, GVW13, GGH$^+$13b, GKP$^+$13b, GGH$^+$13a]. Details of this are provided in Section 1.1.3.

Our Goal. As these examples illustrate, extending the scope of functional encryption to address functions defined over multiple ciphertexts can be highly beneficial. In short, it could provide a non-interactive method to compute n-ary functions on encrypted inputs (possibly by different parties), analogously to interactive multi-party secure computations defined over multiple inputs held by n different parties.

Extending functional encryption to address the multi-input setting is the focus of this work.

1.1 This paper

This paper is dedicated to the study of multi-input functional encryption, starting with formalizations of security. We provide both feasibility results and negative results with respect to different definitions of security. Following the single-input setting, we consider two notions of security, namely, indistinguishability-based security (or IND security for short) and simulation-based security (or SIM security for short).

1.1.1 Indistinguishability-based Security

We start by considering the notion of indistinguishability-based security for functional encryption for n-ary functions: Informally speaking, in IND security for MI-FE, we consider a game between a judge and an adversary. First, the judge generates the master secret key MSK, n encryption keys $\{\mathsf{EK}_1, \ldots, \mathsf{EK}_n\}$ and gives to the adversary a subset of the encryption keys (chosen by the adversary). Then the adversary can request any number of secret keys SK_f for functions f of her choice. Next, the adversary declares two "challenge vectors" $\vec{X}^0$ and $\vec{X}^1$, where every $X_i^b \in \vec{X}^b$ is a set of plaintexts $\{x_{i,1}^b, \ldots, x_{i,n}^b\}$. The judge chooses a bit b at random, and for each $j \in [n]$, the judge encrypts every element $x_{i,j}^b$ of X_i^b (for every i) using encryption key EK_j to obtain a tuple of "challenge ciphertexts" $\vec{\mathsf{CT}}$, which is given to the adversary. After this, the adversary can again request any number of secret keys SK_f for functions f of her choice. Finally, the adversary has to guess the bit b that the judge chose.

If the adversary has requested a secret key for any function f such that there exist *splitting* input vectors $\vec{y}^0$ and $\vec{y}^1$ that satisfy the following two properties:

1. For every $j \in [n]$, either $\exists i$ such that $y_j^b \in X_i^b$ or the adversary has EK_j, and
2. $f(\vec{y}^0) \neq f(\vec{y}^1)$,

then the adversary loses the game – because the legitimate functionalities that he has access to already allow him to distinguish between the scenario where $b = 0$ and $b = 1$. If the adversary never queries a secret key for such a function but nevertheless guesses b correctly, we say that she wins. The IND security definition requires that the adversary's probability of winning be at most negligibly greater than $\frac{1}{2}$.

This definition generalizes the indistinguishability-based definition of (single-input) functional encryption, which was historically the first security notion considered for functional encryption [SW05]. Informally speaking, this definition captures an information-theoretic flavor of

security, where the adversary should not learn anything beyond what is information-theoretically revealed by the function outputs it can obtain.

With regards to IND-secure MI-FE, we obtain the following results:

IND-secure MI-FE from Indistinguishability Obfuscation. Assume the existence of an indistinguishability obfuscator [BGI+01] for general circuits (the first candidate construction for the same was recently put forward by [GGH+13a]) and one-way functions, we provide a construction for IND-secure MI-FE for general circuits for any polynomial-size challenge vectors, with any subset of encryption keys given to the adversary. Furthermore, our construction has security when the adversary can obtain any *unbounded* polynomial number of secret keys SK_f. We prove the security in the selective model, where where the adversary must begin by declaring the challenge vectors. By using complexity leveraging (and thereby assuming sub-exponentially secure indistinguishability obfuscation and sub-exponentially secure one-way functions, we can achieve full security in a standard manner.

***Compact* IND-secure MI-FE from Differing-Inputs Obfuscation.** Our first construction only supports challenge vectors with an a priori fixed (polynomial) size q. In particular, the size of the encryption keys and ciphertexts in the scheme grows with q. Towards this end, assuming the existence of the stronger notion of differing-inputs obfuscation [BGI+01] and one-way functions, we provide a second construction for IND secure MI-FE with "compact" keys and ciphertexts, i.e., the size of the keys and ciphertexts in the scheme is *independent* of q. Further, we directly prove *full* security of our scheme against adversaries that know any subset of encryption keys and an unbounded polynomial number of secret keys SK_f.

IND-secure MI-FE implies Indistinguishability Obfuscation. Finally, we show that the existence of IND-secure MI-FE for general circuits implies the existence of an indistinguishability obfuscator for general circuits, even when:

1. The MI-FE scheme is only secure against adversaries that can obtain a *single* secret key.
2. The adversary does not know any encryption keys, i.e., the MI-FE scheme is a *secret-key* scheme.

This stands in stark contrast to the single-input setting, where [SS10] showed how to obtain single-key secure (single input) functional encryption for all circuits, under only the assumption that public-key encryption exists. Indeed, further research in single-key security for functional encryption has largely focused on efficiency issues [GKP+13b, GKP+13a] such as succinctness of ciphertexts, that enable new applications. In the setting of multi-input security, in contrast, even single key security must rely on the existence of indistinguishability obfuscation.

1.1.2 Simulation-based security

In simulation-based security, informally speaking, we require that every adversary can be simulated using only oracle access to the functions f for which the adversary obtains secret keys, even when it can obtain a set of "challenge" ciphertexts corresponding to unknown plaintexts – about which the simulator can only learn information by querying the function f at these unknown plaintexts. We highlight two natural settings for the study of SIM-secure MI-FE: (1) the setting where an adversary has access to an encryption key (analogous to the public-key setting), and (2) the setting where the adversary does not have access to any encryption keys (analogous to the secret key setting). The security guarantees which are achievable in these settings will be vastly different as illustrated below.

Several works [BSW11, AGVW13, BO13, CIJ+13] have shown limitations on parameters with respect to which SIM security can be achieved for single-input functional encryption. For

multi-input functional encryption, due to the connection to obfuscation discussed above, the situation for SIM security is more problematic. We provide the following results for SIM-secure MI-FE:

SIM-secure MI-FE implies Virtual Black-Box Obfuscation. We first show that SIM-secure MI-FE implies virtual black-box (VBB) obfuscation in various settings. Specifically, we show:

1. If there exists a *secret-key* MI-FE scheme for general circuits that achieves SIM security against adversaries that request: (a) a *single* key for a general function f and (b) a set of challenge ciphertexts that can (informally speaking) form a *super-polynomial* number of potential inputs to f, then VBB obfuscation must be possible for general circuits.
2. If there exists an MI-FE scheme for 2-ary functions that achieves SIM security against adversaries that request: (a) a *single* key for a 2-ary function, and (b) *one* of the two encryption keys and *one* challenge ciphertext, then VBB obfuscation must be possible for general circuits.

Since VBB obfuscation is known to be impossible for general circuits [BGI+01], this yields us impossibility results for SIM-secure MI-FE beyond those known in the single-input setting. See Section 6 for details.

SIM-secure Secret-Key MI-FE against Unbounded Collusions. In light of these negative results, the only hope for obtaining a positive result lies in a situation where: (a) *no* encryption keys are given to the adversary, and (b) the challenge ciphertexts given to the adversary can only form a *polynomial* number of potential inputs to valid functions.

Towards this end, assuming one-way functions and indistinguishability obfuscation, for any fixed polynomial bound q on the size of challenge plaintexts, we give a construction for SIM-secure secret-key MI-FE for general circuits against adversaries that can obtain an unbounded polynomial number of secret keys SK_f *after* obtaining the challenge ciphertexts. The size of the encryption keys and ciphertexts in this scheme grows with q.

We also provide another construction based on one-way functions and differing-inputs obfuscation that achieves the same security guarantees as above. The encryption keys and ciphertexts in this scheme are "compact", i.e., their sizes are independent of q.

1.1.3 Extensions and Applications

MI-FE for Randomized Functions. Very recently, Goyal et al. [GJKS13] first studied the question of constructing single-input functional encryption schemes for *randomized* functionalities. By building on their techniques, we show how to extend our positive results to handle general n-ary randomized functionalities. In particular, this allows us to obtain a non-interactive computationally differentially private mechanism, as discussed earlier.

MI-FE for Turing Machines. The problem of single-input functional encryption for turing machines was first studied by Goldwasser et al. [GKP+13a]. Very recently, Boyle et al. [BCP13] and Ananth et al. [ABG+13] provide constructions of single-input functional encryption for turing machines against an unbounded polynomial number of key queries. We observe that their techniques can be leveraged to extend our results to MI-FE for turing machines, thereby achieving *input-specific running times*. The resulting construction would inherit from these works the underlying assumptions of differing-inputs obfuscation, succinct non-interactive argument of knowledge (SNARK) [BCCT12], fully-homomorphic encryption [Gen09] and collision-resistant hash functions. We omit the details from this manuscript and refer the reader to [ABG+13, BCP13].

Hierarchical MI-FE. The notion of hierarchical identity-based and attribute-based encryption is well studied in the literature (see e.g., [GS02, BW06, LOS+10]). In the context of (single-input) functional encryption, this problem is stated as follows: We require that the owner of a secret key SK_f can derive new keys corresponding to any function g that can defined as a composition of f' on f (i.e., $f' \circ f$) for some function f'.

Recently, [ABG+13] observe that the construction [GGH+13a] is already flexible enough to yield a hierarchical (single-input) functional encryption scheme. We note that the same ideas carry over to our constructions of MI-FE. We refer the reader to [ABG+13] for details.

Multi-Client Delegation of Computation. Here we briefly discuss how an MI-FE scheme provides a solution for multi-client delegation of computation. We follow the approach of Parno et al. [PRV12], adapted to the multi-client setting. Given an MI-FE scheme, the clients first participate in a pre-processing phase where they jointly compute two pairs of master secret and encryption keys $(\mathsf{MSK}_1, \mathsf{EK}_1)$, $(\mathsf{MSK}_2, \mathsf{EK}_2)$ and random values (r_1, r_2). Let f be the function that the clients wish to delegate. The clients use MSK_1 to compute a secret key SK_g for a function g that takes as input n tuples $(x_1, r), \ldots, (x_n, r)$ and outputs r if $f(x_1, \ldots, x_n) = 1$. Similarly, the clients use MSK_2 to compute a secret key $\bar{\mathsf{SK}}_g$ for the function $\bar{g}$ that is the same as g except that it outputs r if $f(x_1, \ldots, x_n) = 0$. While these are computationally expensive operations, note that this phase is executed only *once*. The keys SK_g and $\bar{\mathsf{SK}}_g$ are sent over to the worker.

Later, in an "online" phase, when the clients wish to compute f on a set of inputs $x_1, \ldots, x_n$, each client C_i sends over encryption of (x_1, r_1) under key MPK_1 and (x_1, r_2) under MPK_2 to the worker. Now, from the properties of the MI-FE scheme, it follows that if $f(x_1, \ldots, x_n)$, then the server would obtain r_1 using SK_g and $\perp$ using $\bar{\mathsf{SK}}_g$ and no information about r_2 (and vice-versa, if $f(x_1, \ldots, x_n) = 0$). Thus, r_1 provides a proof of the fact that the function output is 1.[1]

The main advantage of this approach is that the online phase is *non-intractive*: each client can execute the online phase independently of the other clients, without any interaction.

1.1.4 Our Techniques

We have several results in this work, but to provide a flavor of the kind of difficulties that arise in the MI-FE setting, we now discuss some of the issues that we deal with in the context of our positive result for IND-secure MI-FE. (We note that similar issues arise in our positive results for SIM-secure MI-FE.)

The starting point for our construction and analysis is the recent single-input functional encryption scheme for general circuits based on indistinguishability obfuscation due to [GGH+13a]. However, the central issue that we must deal with is one that does not arise in their context: Recall that in the indistinguishability security game, the adversary is allowed to get secret keys for any function f, as long as this function does not "split" the challenge vectors $\vec{X}^0$ and $\vec{X}^1$. That is, as long as it is *not* the case that there exist vectors of plaintexts $\vec{x}^0$ and $\vec{x}^1$ where for every $i \in [n]$, either there exists j such that $x_i^b \in X_j^b$ or the adversary has EK_i, such that $f(x^0) \neq f(x^1)$. A crucial point here is what happens for an index i where the adversary does *not* have EK_i. Let us consider an example with a 3-ary function, where the adversary has EK_1, but neither EK_2 nor EK_3.

Suppose the challenge ciphertexts $(\mathsf{CT}_1, \mathsf{CT}_2, \mathsf{CT}_3)$ are encryptions of either (y_1^0, y_2^0, y_3^0) or (y_1^1, y_2^1, y_3^1). Now, any function f that the adversary queries is required to be such that $f(\cdot, y_2^0, y_3^0) \equiv f(\cdot, y_2^1, y_3^1)$ and $f(y_1^0, y_2^0, y_3^0) = f(y_1^1, y_2^1, y_3^1)$. However, there may exist an input plaintext (say) z such that $f(y_1^0, y_2^0, z) \neq f(y_1^1, y_2^1, z)$. This is not "supposed" to be a problem because the adversary does not have EK_3, and therefore it cannot actually query f with z as its third argument.

[1] We note that this solution easily extends to functions with multi-bit outputs. See [PRV12] for details.

However, in the obfuscation-based approach to functional encryption of [GGH+13a] that we build on, the secret key for f is essentially built on top of an obfuscation of f. Let CT^* denote an encryption of z w.r.t. EK_3. Then, informally speaking, in one of our hybrid experiments, we will need to move from an obfuscation that on input $(\mathsf{CT}_1, \mathsf{CT}_2, \mathsf{CT}^*)$ would yield the output $f(y_1^0, y_2^0, z)$ to another obfuscation that on the same input would yield the output $f(y_1^1, y_2^1, z)$. Again, while an adversary may not be able explicitly perform such a decryption query, since we are building upon *indistinguishability obfuscation* – which only guarantees that obfuscations of circuits that implement *identical* functions are indistinguishable – such a hybrid change would not be indistinguishable since we know that $f(y_1^0, y_2^0, z) \neq f(y_1^1, y_2^1, z)$ are not identical. (We remark that we must address this issue even when using differing-inputs obfuscation in order to obtain a formal contradiction.)

Solving this problem is the core technical aspect of our constructions and their analysis. At a very high level, we address this problem by introducing a new "flag" value that can change the nature of the function f that we are obfuscating to "disable" all plaintexts except for the ones that are in the challenge vectors. We describe the details and our analysis in Section 4.

1.2 Related Works

Single-input Functional Encryption. The notion of (single-input) functional encryption was developed in a sequence of works [SW05, GPSW06, BW07, KSW08, LOS+10, BSW11, O'N10]. For general functions, [SS10] first showed how to obtain single-key SIM-secure FE based on standard public-key encryption. Gorbunov et al [GVW12] showed how to obtain SIM-secure FE for general circuits for a polynomially bounded number of (non-adaptive) key queries, based on public-key encryption and pseudorandom generators in $\mathbf{NC}^1$. Goldwasser et al. [GKP+13b] improved this result to obtain constructions with "compact" ciphertexts based on sub-exponential learning with errors assumption. Garg et al. [GGH+13a] construct an IND-secure FE scheme based on indistinguishability obfuscation and one-way functions, that supports an unbounded polynomial number of ciphertexts and key queries. Combining their result with [CIJ+13], one can obtain SIM-secure FE for general circuits supporting an unbounded number of (adaptive) key queries.

Goldwasser et al. [GKP+13a] give a construction of an FE scheme for turing machines based on extractable witness encryption [GGSW13] and SNARK [BCCT12]. Recently, the works of Boyle et al. [BCP13] and Ananth et al. [ABG+13] provide constructions of functional encryption for turing machines, supporting an unbounded number of key queries. Both of these results rely on the notion of differing-inputs obfuscation, introduced by Barak et al. [BGI+01] (and some other assumptions; see Section 1.1.3). We note that our usage of differing-inputs obfuscation is very similar to [BCP13, ABG+13].

Order-Preserving Encryption. The notion of order-preserving encryption was introduced by Boldyreva et al. [BCLO09]. Very roughly, in an order-preserving encryption scheme, for any two plaintexts x_1 and x_2 such that $x_1 > x_2$, the encryptions of x_1 and x_2 must also satisfy the same order relationship. Thus, given two ciphertexts CT_1 and CT_2, one can simply compare them to (publicly) determine the order relationship between their underlying plaintexts.

Positive results for order-preserving encryption were given by [BCLO09, BCO11]. These results, however, achieve very weak security guarantees (in particular, they show that an order-preserving encryption scheme cannot achieve IND security). We note that one can cast the problem of computing order relationships between (encrypted) plaintexts as multi-input function encryption for comparison functionality. Specifically, instead of requiring that ciphertexts obey the same order relationship as their underlying plaintexts, we can now release secret keys to enable the computation of order relationship between encrypted plaintexts. This allows us to achieve IND security as well as SIM security, both of which provide much stronger guarantees

than [BCLO09, BCO11]. Indeed, achieving stronger security guarantees in this context was left as an open problem by [BCLO09, BCO11].

Property-Preserving Encryption. Recently, Pandey and Rouselakis [PR12] studied the problem of property-preserving encryption as a generalization of order-preserving encryption. As above, we note that this problem can be viewed as a multi-input functional encryption, where the function family is determined by the class of properties that one wishes to support. Again, we note that the security definitions considered in [PR12] are weaker than what we consider in this work. In particular, this is because we do not require the ciphertexts to satisfy the same property as their underlying plaintexts; instead in our setting, given a secret key SK_f for a property f, one can test f on the plaintexts via a joint decryption of the corresponding ciphertexts.

1.3 Organization

The rest of this paper is organized as follows. We start by presenting our definitions for multi-input functional encryption in Section 2. Next, in Section 3, we recall the definitions for various cryptographic primitives used in our constructions. We then present our constructions for multi-input functional encryption in Section 4 and Section 5. In Section 6, we show how to construct general obfuscation from multi-input functional encryption and also provide impossibility results for SIM-secure MI-FE. Finally, we discuss how to extend our positive results to handle randomized functionalities in Section 7.

2 Multi-Input Functional Encryption

In this work, we study functional encryption for n-ary functions, where $n > 1$ (and in general, a polynomial in the security parameter). In other words, we are interested in encryption schemes where the owner of a "master" secret key can generate special keys SK_f that allow the computation of $f(x_1, \ldots, x_n)$ from n ciphertexts $\mathsf{CT}_1, \ldots, \mathsf{CT}_n$ corresponding to messages $x_1, \ldots, x_n$, respectively. We refer to such an encryption scheme as *multi-input* functional encryption. Analogously, we will refer to the existing notion of functional encryption (that only considers single-ary functions) as *single-input* functional encryption.

Intuitively, while single-input functional encryption can be viewed as a specific (non-interactive) way of performing two-party computation, our setting of multi-input functional encryption captures *multiparty* computation. Going forward with this analogy, we are interested in modeling the general scenario where the n input ciphertexts are computed by n *different* parties. This raises the following two important questions:

1. Do the parties (i.e., the encryptors) share the *same* encryption key or do they use *different* encryption keys EK_i to compute input ciphertexts CT_i.
2. Are the encryption keys secret or public?

As we shall see, these questions have important bearing on the security guarantees that can be achieved for multi-input functional encryption.

Towards that end, we present a general, unified syntax and security definitions for multi-input functional encryption. We consider encryption systems with n encryption keys, some of which may be public, while the rest are secret. When all of the encryption keys are public, then this represents the "public-key" setting, while when all the encryption keys are secret, then this represents the "secret-key" setting. Looking ahead, we remark that our modeling allows us to capture the intermediary cases between these two extremes that are interesting from the viewpoint of the security guarantees possible.

The rest of this section is organized as follows. We first present the syntax and correctness requirements for multi-input FE in Section 2.1). Then, in Section 2.2, we present our security definitions for multi-input FE.

2.1 Syntax

Throughout the paper, we denote the security parameter by k. Let $\mathcal{X} = \{\mathcal{X}_k\}_{k\in\mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_k\}_{k\in\mathbb{N}}$ be ensembles where each $\mathcal{X}_k$ and $\mathcal{Y}_k$ is a finite set. Let $\mathcal{F} = \{\mathcal{F}_k\}_{k\in\mathbb{N}}$ be an ensemble where each $\mathcal{F}_k$ is a finite collection of n-ary functions. Each function $f \in \mathcal{F}_k$ takes as input n strings $x_1, \ldots, x_n$, where each $x_i \in \mathcal{X}_k$ and outputs $f(x_1, \ldots, x_n) \in \mathcal{Y}_k$.

A multi-input functional encryption scheme $\mathcal{FE}$ for $\mathcal{F}$ consists of four algorithms ($\mathsf{FE.Setup}$, $\mathsf{FE.Enc}$, $\mathsf{FE.Keygen}$, $\mathsf{FE.Dec}$) described below.

- **Setup** $\mathsf{FE.Setup}(1^k, n)$ is a PPT algorithm that takes as input the security parameter k and the function arity n. It outputs n encryption keys $\mathsf{EK}_1, \ldots, \mathsf{EK}_n$ and a master secret key MSK.
- **Encryption** $\mathsf{FE.Enc}(\mathsf{EK}, x)$ is a PPT algorithm that takes as input an encryption key $\mathsf{EK}_i \in (\mathsf{EK}_1, \ldots, \mathsf{EK}_n)$ and an input message $x \in \mathcal{X}_k$ and outputs a ciphertext CT.

  In the case where all of the encryption keys EK_i are the same, we assume that each ciphertext CT has an associated label i to denote that the encrypted plaintext constitutes an i'th input to a function $f \in \mathcal{F}_k$. For convenience of notation, we omit the labels from the explicit description of the ciphertexts. In particular, note that when EK_i's are *distinct*, the index of the encryption key EK_i used to compute CT implicitly denotes that the plaintext encrypted in CT constitutes an i'th input to f, and thus no explicit label is necessary.
- **Key Generation** $\mathsf{FE.Keygen}(\mathsf{MSK}, f)$ is a PPT algorithm that takes as input the master secret key MSK and an n-ary function $f \in \mathcal{F}_k$ and outputs a corresponding secret key SK_f.
- **Decryption** $\mathsf{FE.Dec}(\mathsf{SK}_f, \mathsf{CT}_1, \ldots, \mathsf{CT}_n)$ is a deterministic algorithm that takes as input a secret key SK_f and n ciphertexts $\mathsf{CT}_i, \ldots, \mathsf{CT}_n$ and outputs a string $y \in \mathcal{Y}_k$.

Definition 1 (Correctness)**.** *A multi-input functional encryption scheme $\mathcal{FE}$ for $\mathcal{F}$ is* correct *if for all $f \in \mathcal{F}_k$ and all $(x_1, \ldots, x_n) \in \mathcal{X}_k^n$:*

$$\Pr\left[\begin{array}{l}(\vec{\mathsf{EK}}, \mathsf{MSK}) \leftarrow \mathsf{FE.Setup}(1^k)\,;\ \mathsf{SK}_f \leftarrow \mathsf{FE.Keygen}(\mathsf{MSK}, f)\,; \\ \qquad \mathsf{FE.Dec}\left(\mathsf{SK}_f, \mathsf{FE.Enc}\left(\mathsf{EK}_1, x_1\right), \ldots, \mathsf{FE.Enc}\left(\mathsf{EK}_n, x_n\right)\right) \neq f(x_1, \ldots, x_n)\end{array}\right] = \mathsf{negl}(k)$$

where the probability is taken over the coins of $\mathsf{FE.Setup}$*,* $\mathsf{FE.Keygen}$ *and* $\mathsf{FE.Enc}$*.*

2.2 Security for Multi-Input Functional Encryption

We now present our security definitions for multi-input functional encryption. Following the literature on single-input FE, we consider two notions of security, namely, indistinguishability-based security (or IND-security, in short) and simulation-based security (or SIM-security, in short).

Notation. We start by introducing some notation that is used in our security definitions. Let N denote the set of positive integers $\{1, \ldots, n\}$ where n denotes the arity of functions. For any two sets $S = \{s_0, \ldots, s_{|S|}\}$ and $I = \{i_1, \ldots, i_{|I|}\}$ such that $|I| \leq |S|$, we let S_I denote the subset $\{s_i\}_{i\in I}$ of the set S. Throughout the text, we use the vector and set notation interchangeably, as per convenience. For simplicity of notation, we omit explicit reference to auxiliary input to the adversary from our definitions.

2.2.1 Indistinguishability-based Security

Here we present an indistinguishability-based security definition for multi-input FE.

Intuition. We start by giving an overview of the main ideas behind our indistinguishability-based security definition. To convey the core ideas, it suffices to consider the case of 2-ary functions. We will assume familiarity with the security definitions for single-input FE.

Let us start by considering the natural extension of *public-key* single-input FE to the two-input setting. That is, suppose there are two public encryption keys EK_1, EK_2 that are used to create ciphertexts of first inputs and second inputs, respectively, for 2-ary functions. Let us investigate what security can be achieved for *one* pair of challenge message tuples (x_1^0, x_2^0), (x_1^1, x_2^1) for the simplified case where the adversary makes secret key queries after receiving the challenge ciphertexts.

Suppose that the adversary queries secret keys for functions $\{f\}$. Now, recall that the IND-security definition in the single-input case guarantees that an adversary cannot differentiate between encryptions of x^0 and x^1 as long as $f(x^0) = f(x^1)$ for every $f \in \{f\}$. We note, however, that an analogous security guarantee cannot be achieved in the multi-input setting. That is, restricting the functions $\{f\}$ to be such that $f(x_1^0, x_2^0) = f(x_1^1, x_2^1)$ is *not* enough since an adversary who knows both the encryption keys can create its own ciphertexts w.r.t. each encryption key. Then, by using the secret key corresponding to function f, it can learn additional values $\{f(x_1^b, \cdot)\}$ and $\{f(\cdot, x_2^b)\}$, where b is the challenge bit. In particular, if, for example, there exists an input x^* such that $f(x_1^0, x^*) \neq f(x_1^1, x^*)$, then the adversary can learn the challenge bit b! Therefore, we must enforce additional restrictions on the query functions f. Specifically, we must require that $f(x_1^0, x') = f(x_1^1, x')$ for *every* input x' in the domain (and similarly $f(x', x_2^0) = f(x', x_2^1)$). Note that this restriction "grows" with the arity n of the functions.

Let us now consider the secret-key case, where all the encryption keys are secret. In this case, for the above example, it suffices to require that $f(x_1^0, x_2^0) = f(x_1^1, x_2^1)$ since the adversary cannot create its own ciphertexts. Observe, however, that when there are *multiple* challenge messages, then an adversary can learn function evaluations over different "combinations" of challenge messages. In particular, if there are q challenge messages per encryption key, then the adversary can learn q^2 output values for every f. Then, we must enforce that for every $i \in [q^2]$, the i'th output value y_i^0 when challenge bit $b = 0$ is *equal* to the output value y_i^1 when the challenge bit $b = 1$.

The security guarantees in the public-key and the secret-key settings as discussed above are vastly different. In general, we observe that the *more* the number of encryption keys that are public, the *smaller* the class of functions that can be supported by the definition. Bellow, we present a unified definition that simultaneously captures the extreme cases of public-key and secret-key settings as well as all the "in between" cases.

Compatible Functions and Input Plaintexts. To facilitate the presentation of our IND security definition, we first introduce the following notion:

Definition 2 (I-Compatibility)**.** *Let $\{f\}$ be any set of functions $f \in \mathcal{F}_k$. Let $\mathsf{N} = \{1, \ldots, n\}$ and $\mathrm{I} \subseteq \mathsf{N}$. Let $\vec{X}^0$ and $\vec{X}^1$ be a pair of input vectors, where $\vec{X}^b = \left\{x_{1,j}^b, \ldots, x_{n,j}^b\right\}_{j=1}^q$. We say that $\mathcal{F}$ and $(\vec{X}^0, \vec{X}^1)$ are* I-compatible *if they satisfy the following property:*

- *For every $f \in \{f\}$, every $\mathrm{I}' = \{i_1, \ldots, i_t\} \subseteq \mathrm{I} \cup \emptyset$, every $j_1, \ldots, j_{n-t} \in [q]$, and every $x'_{i_1}, \ldots, x'_{i_t} \in \mathcal{X}_k$,*

$$f\left(\left\langle x_{i_1,j_1}^0, \ldots, x_{i_{n-t'},j_{n-t}}^0, x'_{i_1}, \ldots, x'_{i_t}\right\rangle\right) = f\left(\left\langle x_{i_1,j_1}^1, \ldots, x_{i_{n-t},j_{n-t}}^1, x'_{i_1}, \ldots, x'_{i_t}\right\rangle\right),$$

  where $\langle y_{i_1}, \ldots, y_{i_n}\rangle$ denotes a permutation of the values $y_{i_1}, \ldots, y_{i_n}$ such that the value y_{i_j} is mapped to the ℓ'th location if y_{i_j} is the ℓ'th input (out of n inputs) to f.

IND-secure MI-FE. Our security definition is parameterized by two variables t and q, where t denotes the number of encryption keys known to the adversary, and q denotes the number of challenge messages per encryption key. Thus, in total, the adversary is allowed to make $\mathrm{Q} = q \cdot n$ number of challenge message queries. We are now ready to present our formal definition for (t, q)-IND-secure multi-input functional encryption.

Definition 3 (Indistinguishability-based security)**.** *We say that a multi-input functional encryption scheme $\mathcal{FE}$ for for n-ary functions $\mathcal{F}$ is* (t, q)-IND-secure *if for every PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$, the advantage of $\mathcal{A}$ defined as*

$$\mathsf{Adv}^{\mathcal{FE},\mathsf{IND}}_{\mathcal{A}}(1^k) = \left|\Pr[\mathsf{IND}^{\mathcal{FE}}_{\mathcal{A}}(1^k) = 1] - \frac{1}{2}\right|$$

is $\mathsf{negl}(k)$*, where:*

Experiment $\mathsf{IND}^{\mathcal{FE}}_{\mathcal{A}}(1^k)$*:*
$(\mathrm{I}, \mathsf{st}_0) \leftarrow \mathcal{A}_0(1^k)$ *where* $|\mathrm{I}| = t$
$(\vec{\mathsf{EK}}, \mathsf{MSK}) \leftarrow \mathsf{FE.Setup}(1^k)$
$(\vec{X}^0, \vec{X}^1, \mathsf{st}_1) \leftarrow \mathcal{A}_1^{\mathsf{FE.Keygen}(\mathsf{MSK},\cdot)}(\mathsf{st}_0, \vec{\mathsf{EK}}_{\mathrm{I}})$ *where* $\vec{X}^\ell = \left\{x^\ell_{1,j}, \ldots, x^\ell_{n,j}\right\}_{j=1}^{q}$
$b \leftarrow \{0,1\}$; $\mathsf{CT}_{i,j} \leftarrow \mathsf{FE.Enc}(\mathsf{EK}_i, x^b_{i,j})$ $\forall i \in [n]$, $j \in [q]$
$b' \leftarrow \mathcal{A}_2^{\mathsf{FE.Keygen}(\mathsf{MSK},\cdot)}(\mathsf{st}_1, \vec{\mathsf{CT}})$
Output: $(b = b')$

In the above experiment, we require:

- *Let $\{f\}$ denote the entire set of key queries made by $\mathcal{A}_1$. Then, the challenge message vectors $\vec{X}_0$ and $\vec{X}_1$ chosen by $\mathcal{A}_1$ must be* I*-compatible with $\{f\}$.*
- *The key queries $\{g\}$ made by $\mathcal{A}_2$ must be* I*-compatible with $\vec{X}^0$ and $\vec{X}^1$.*

Selective Security. We also consider selective *indistinguishability*-based security for multi-input functional encryption. Formally, (t, q)-sel-IND-security is defined in the same manner as Definition 3, except that the adversary $\mathcal{A}_1$ is required to choose the challenge message vectors $\vec{X}^0$, $\vec{X}^1$ *before* the evaluation keys $\vec{\mathsf{EK}}$ and the master secret key MSK are chosen by the challenger. We omit the formal definition to avoid repetition.

2.2.2 Simulation-based Security

Here we present a simulation-based security definition for multi-input FE. We consider the case where the adversary makes key queries *after* choosing the challenge messages. That is, we only consider *adaptive* key queries. The "opposite" case where the adversary makes key queries *before* choosing the challenge messages (i..e, *non-adaptive* key queries) is discussed in Section D.

Our definition extends the simulation-based security definition for single-input FE that supports adaptive key queries[BSW11, O'N10, BO13, CIJ+13]. In particular, we present a general definition that models both black-box and non-black-box simulation.

Intuition. We start by giving an overview of the main ideas behind our simulation-based security definition. To convey the core ideas, it suffices to consider the case of 2-ary functions. Let us start by considering the natural extension of *public-key* single-input FE to the two-input setting. That is, suppose there are two public encryption keys EK_1, EK_2 that are used to create ciphertexts of first inputs and second inputs, respectively, for 2-ary functions. Let us investigate what security can be achieved for *one* challenge message tuple (x_1, x_2).

Suppose that the adversary queries secret keys for functions $\{f\}$. Now, recall that the SIM-security definition in the single-input case guarantees that for every $f \in \{f\}$, an adversary cannot learn more than $f(x)$ when x is the challenge message. We note, however, that an analogous security guarantee cannot be achieved in the multi-input setting. Indeed, an adversary who knows both the encryption keys can create its own ciphertexts w.r.t. each encryption key. Then, by using the secret key corresponding to function f, it can learn additional values $\{f(x_1, \cdot)\}$ and $\{f(\cdot, x_2)\}$. Thus, we must allow for the ideal world adversary, aka simulator, to learn the same information.

In the secret-key case, however, since all of the encryption keys are secret, the SIM-security definition for single-input FE indeed extends in a natural manner to the multi-input setting. We stress, however, that when there are multiple challenge messages, we must take into account the fact that adversary can learn function evaluations over all possible "combinations" of challenge messages. Our definition presented below formalizes this intuition.

SIM-secure MI-FE. Similar to the IND-security case, our definition is parameterized by variables t and q as defined earlier. We now formally define (t, q)-SIM-secure multi-input functional encryption.

Definition 4 (Simulation-based Security)**.** *We say that a functional encryption scheme $\mathcal{FE}$ for n-ary functions $\mathcal{F}$ is* (t, q)-SIM-secure *if for every PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$, there exists a PPT simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2)$ such that the outputs of the following two experiments are computationally indistinguishable:*

Experiment $\mathsf{REAL}_{\mathcal{A}}^{\mathcal{FE}}(1^k)$*:*
$(\mathrm{I}, \mathsf{st}_0) \leftarrow \mathcal{A}_0(1^k)$ *where* $|\mathrm{I}| = t$
$(\vec{\mathsf{EK}}, \mathsf{MSK}) \leftarrow \mathsf{FE.Setup}(1^k)$
$(\mathcal{M}, \mathsf{st}_1) \leftarrow \mathcal{A}_1(\mathsf{st}_0, \vec{\mathsf{EK}}_{\mathrm{I}})$
$\vec{X} \leftarrow \mathcal{M}$ *where* $\vec{X} = \{x_{1,j}, \ldots, x_{n,j}\}_{j=1}^{q}$
$\mathsf{CT}_{i,j} \leftarrow \mathsf{FE.Enc}(\mathsf{EK}_i, x_{i,j})\ \forall i \in [n],\ j \in [q]$
$\alpha \leftarrow \mathcal{A}_2^{\mathsf{FE.Keygen}(\mathsf{MSK}, \cdot)}(\vec{\mathsf{CT}}, \mathsf{st}_1)$
Output: $(\mathrm{I}, \mathcal{M}, \vec{X}, \{f\}, \alpha)$

Experiment $\mathsf{IDEAL}_{\mathcal{S}}^{\mathcal{FE}}(1^k)$*:*
$(\mathrm{I}, \mathsf{st}_0) \leftarrow \mathcal{S}_0(1^k)$
$(\mathcal{M}, \mathsf{st}_1) \leftarrow \mathcal{S}_1(\mathsf{st}_0)$
$\alpha \leftarrow \mathcal{S}_2^{\mathsf{TP}(\mathcal{M}, \cdot, \cdot)}(\mathsf{st}_1)$
Output: $(\mathrm{I}, \mathcal{M}, \vec{X}, \{g\}, \alpha)$

where the oracle $\mathsf{TP}(\mathcal{M}, \cdot, \cdot)$ denotes the ideal world trusted party, $\{f\}$ denotes the set of queries of $\mathcal{A}_2$ to FE.Keygen *and $\{g\}$ denotes the set of functions appearing in the queries of $\mathcal{S}_2$ to* TP*. Given the message distribution $\mathcal{M}$,* TP *first samples a message vector $\vec{X} \leftarrow M$, where $\vec{X} = \{x_{1,j}, \ldots, x_{n,j}\}_{j=1}^{q}$. It then accepts queries of the form $\left(g, (j_1, \ldots, j_{n-p}), \left(x'_{i'_1}, \ldots, x'_{i'_p}\right)\right)$ where $p \leq t$, $\{i'_1, \ldots, i'_p\} \subseteq \mathrm{I} \cup \emptyset$ and $x'_{i'_1}, \ldots, x'_{i'_p} \in \mathcal{X}_k$. On receiving such a query,* TP *outputs:*

$$g\left(\left\langle x_{i_1,j_1}, \ldots, x_{i_{n-p},j_{n-p}}, x'_{i'_1}, \ldots, x'_{i'_p} \right\rangle\right),$$

where $\langle y_{i_1}, \ldots, y_{i_n} \rangle$ denotes a permutation of the values $y_{i_1}, \ldots, y_{i_n}$ such that the value y_{i_j} is mapped to the ℓ'th location if y_{i_j} is the ℓ'th input (out of n inputs) to g.

Remark 5 (On Queries to the Trusted Party)**.** *Note that when $t = 0$, then given the challenge ciphertexts $\vec{\mathsf{CT}}$, intuitively, the real adversary can only compute values $\mathsf{FE.Dec}(\mathsf{SK}_f, \mathsf{CT}_{1,j_1}, \ldots, \mathsf{CT}_{n,j_n})$ for every $j_i \in [q]$, $i \in [n]$. To formalize the intuition that this adversary does not learn anything more than function values $\{f(x_{1,j_1}, \ldots, x_{n,j_n})\}$, we restrict the ideal adversary aka simulator to learn exactly this information.*

However, when $t > 0$, then the real adversary can compute values:

$$\mathsf{FE.Dec}\left(\mathsf{SK}_f, \left\langle \mathsf{CT}_{i_1,j_1}, \ldots, \mathsf{CT}_{i_{n-t},j_{n-t}}, \mathsf{CT}'_{i'_1}, \ldots, \mathsf{CT}'_{i'_t} \right\rangle\right)$$

for ciphertexts $\mathsf{CT}'_{i'_\ell}$ of its choice since it knows the encryption keys $\vec{\mathsf{EK}}_{\mathrm{I}}$. In other words, such an adversary can learn function values of the form $f\left(\langle x_{i_1,j_1}, \ldots, x_{i_{n-t},j_{n-t}}, \cdot, \ldots, \cdot\rangle\right)$. Thus, we must provide the same ability to the simulator as well. Our definition presented above precisely captures this.

Selective Security. We also consider *selective* simulation-based security for multi-input functional encryption. Formally, (t, q)-sel-SIM-security is defined in the same manner as Definition 4, except that in the real world experiment, adversary $\mathcal{A}_1$ chooses the message distribution $\mathcal{M}$ *before* the evaluation keys $\vec{\mathsf{EK}}$ and the master secret key MSK are chosen by the challenger. We omit the formal definition to avoid repetition.

Remark 6 (SIM-security: Secret-key setting)**.** *When $t = 0$,* none *of the encryption keys are known to the adversary. In this "secret-key" setting, there is no difference between $(0, q)$-sel-SIM-security and $(0, q)$-SIM-security.*

3 Preliminaries

Here we present definitions of various cryptographic primitives that are used in our construction of multi-input functional encryption. We assume familiarity with standard semantically-secure public-key encryption and omit its formal definition from this text. Below, we recall the notions of indistinguishability obfuscation, non-interactive witness indistinguishable proof systems and perfectly binding commitment schemes.

3.1 Indistinguishability Obfuscation

Here we recall the notion of indistinguishability obfuscation that was defined by Barak et al. [BGI+01]. Intuitively speaking, we require that for any two circuits C_1 and C_2 that are "functionally equivalent" (i.e., for all inputs x in the domain, $C_1(x) = C_2(x)$), the obfuscation of C_1 must be computationally indistinguishable from the obfuscation of C_2. Below we present the formal definition following the syntax of [GGH+13a].

Definition 7 (Indistinguishability Obfuscation)**.** *A uniform PPT machine $i\mathcal{O}$ is called an indistinguishability obfuscator for a circuit class $\{\mathcal{C}_k\}$ if the following holds:*

- **Correctness:** *For every $k \in \mathbb{N}$, for every $C \in \mathcal{C}_k$, for every input x in the domain of C, we have that*
$$\Pr[C'(x) = C(x) : C' \leftarrow i\mathcal{O}(C)] = 1.$$
- **Indistinguishability:** *For every $k \in \mathbb{N}$, for all pairs of circuits $C_0, C_1 \in \mathcal{C}_k$, if $C_0(x) = C_1(x)$ for all inputs x, then for all PPT adversaries $\mathcal{A}$, we have:*
$$|\Pr[\mathcal{A}(i\mathcal{O}(C_0)) = 1] - \Pr[\mathcal{A}(i\mathcal{O}(C_1)) = 1]| \leq \mathsf{negl}(k).$$

Very recently, Garg et al. [GGH+13a] gave the first candidate construction for an indistinguishability obfuscator $i\mathcal{O}$ for the circuit class $P/poly$. ADD.

Differing-Inputs Obfuscation. We also consider a stronger notion of indistinguishability obfuscation, namely, *differing-inputs obfuscation* that was proposed by Barak et al [BGI+01]. Intuitively speaking, we require that for any two circuits C_1 and C_2 that "appear" to be functionally equivalent to every PPT algorithm (i.e., no PPT algorithm can find an input x s.t. $C_1(x) \neq C_2(x)$), the obfuscation of C_1 must be computationally indistinguishable from the obfuscation of C_2. Alternatively, if a PPT algorithm can distinguish obfuscation of C_1 from obfuscation of C_2, then we can *efficiently* find an input x s.t. $C_1(x) \neq C_2(x)$.

Below, we present the formal definition. We follow the formalism of [ABG+13].

We start by defining the notion of differing-inputs circuit family. Intuitively. a circuit family is said to be a differing-inputs circuits family if there does not exist any PPT adversary that given two circuits, that are sampled from a distribution defined on this circuit family, can find an input x such that both the circuits yield different outputs on x.

Definition 8 (Differing-Inputs Circuit Family)**.** *A circuit family $\mathcal{C}$ associated with a sampler* $\mathsf{Sampler}$ *is said to be a differing-inputs circuit family if for every PPT adversary $\mathcal{A}$, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that:*

$$\Pr\left[C_0(x) \neq C_1(x) \mid (C_0, C_1, z) \leftarrow \mathsf{Sampler}(1^k);\ x \leftarrow \mathcal{A}\left(1^k, C_0, C_1, z\right)\right] \leq \mathsf{negl}(k)$$

Definition 9 (Differing-Inputs Obfuscator)**.** *A PPT machine* $\mathsf{di}\mathcal{O}$ *is called a* differing-inputs obfuscator *for a differing-inputs circuits family $\mathcal{C} = \{C_k\}$ if the following conditions are satisfied:*

- **Correctness:** *For all security parameters $k \in \mathbb{N}$, for all $C \in \mathcal{C}$, for all inputs x, we have that:*
$$\Pr[C'(x) = C(x) \mid C' \leftarrow \mathsf{di}\mathcal{O}(1^k, C)] = 1$$
- **Differing-inputs:** *For any PPT adversary $\mathcal{A}$, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that the following holds: For all security parameters $k \in \mathbb{N}$, for $(C_0, C_1, z) \leftarrow \mathsf{Sampler}(1^k)$, we have that:*
$$\left|\Pr\left[\mathcal{A}\left(\mathsf{di}\mathcal{O}\left(C_1\right)\right) = 1\right] - \Pr\left[\mathcal{A}\left(\mathsf{di}\mathcal{O}\left(C_2\right)\right) = 1\right]\right| \leq \mathsf{negl}(k).$$

3.2 Non-Interactive Proof Systems

In this section, we recall various security notions for non-interactive proof systems. We start by giving the syntax and formal definition of a non-interactive proof system. Next, we give the definition of non-interactive witness-indistinguishable proofs (NIWI). Finally, we give the definition of non-interactive zero-knowledge (NIZK), with simulation-soundness property.

Syntax. Let R be an efficiently computable relation that consists of pairs (x, w), where x is called the statement and w is the witness. Let L denote the language consisting of statements in R. A non-interactive proof system for a language L consists of a setup algorithm CRSGen, a prover algorithm Prove and a verifier algorithm Verify, defined as follows:

- **Setup** $\mathsf{CRSGen}(1^k)$ is a PPT algorithm that takes as input the security parameter k and outputs a common reference string crs.
- **Prover** $\mathsf{Prove}(\mathsf{crs}, x, w)$ is a PPT algorithm that takes as input the common reference string CRS, a statement x along with a witness w. $(x, w) \in R$; if so, it produces a proof string π, else it outputs `fail`.
- **Verifier** $\mathsf{Verify}(\mathsf{crs}, x, \pi)$ is a PPT algorithm that takes as input the common reference string crs and a statement x with a corresponding proof π. It outputs 1 if the proof is valid, and 0 otherwise.

Definition 10 (Non-interactive Proof System)**.** *A* non-interactive proof system *for a language L with a PPT relation R is a tuple of algorithms* $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ *such that the following properties hold:*

- **Perfect Completeness:** *For every $(x, w) \in R$, it holds that*

$$\Pr[\mathsf{Verify}(\mathsf{crs}, x, \mathsf{Prove}(\mathsf{crs}, x, w)) = 1] = 1$$

  where $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^k)$*, and the probability is taken over the coins of* CRSGen*,* Prove *and* Verify*.*

- **Statistical Soundness:** *For every adversary $\mathcal{A}$, it holds that*

$$\Pr[\mathsf{Verify}(\mathsf{crs}, x, \pi) = 1 \wedge x \notin L \mid \mathsf{crs} \leftarrow \mathsf{CRSGen}(1^k); (x, \pi) \leftarrow \mathcal{A}(\mathsf{crs})] = \mathsf{negl}(k)$$

If the soundness property only holds against PPT adversaries, then we call it an argument *system.*

Definition 11 (NIWI)**.** *We say that a non-interactive proof system* $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ *for a language L with a PPT relation R is* witness-indistinguishable *if for any triplet (x, w_0, w_1) such that $(x, w_0) \in R$ and $(x, w_1) \in R$, the distributions* $\{\mathsf{crs}, \mathsf{Prove}(\mathsf{crs}, x, w_0)\}$ *and* $\{\mathsf{crs}, \mathsf{Prove}(\mathsf{crs}, x, w_1)\}$ *are computationally indistinguishable, where* $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^k)$*.*

Definition 12 (NIZK)**.** *A non-interactive proof system* $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ *for a language L with a PPT relation R is said to be* zero knowledge *if there exists a simulator* $\mathsf{Sim} = (\mathsf{Sim.CRSGen}, \mathsf{Sim.Prove})$ *such that for all PPT adversaries $\mathcal{A}$,*

$$\left| \begin{array}{l} \Pr\left[\mathcal{A}^{\mathsf{Prove}(\mathsf{crs},\cdot,\cdot)}(\mathsf{crs}) = 1 \mid \mathsf{crs} \leftarrow \mathsf{CRSGen}\left(1^k\right)\right] \\ \quad - \Pr\left[\mathcal{A}^{\mathcal{S}(\mathsf{crs},\tau,\cdot,\cdot)}(\mathsf{crs}) = 1 \mid (\mathsf{crs}, \tau) \leftarrow \mathsf{Sim.CRSGen}\left(1^k\right)\right] \end{array} \right| = \mathsf{negl}(k)$$

where $\mathcal{S}(\mathsf{crs}, \tau, x, w) = \mathsf{Sim.Prove}(\mathsf{crs}, \tau, x)$ *if $(x, w) \in R$ and outputs* `fail` *otherwise.*

Definition 13 (Simulation soundness)**.** *A* NIZK *proof system* $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ *for a language L with a PPT relation R is said to be* simulation sound *if for all PPT adversaries,*

$$\Pr\left[\begin{array}{l} (x^*, \pi^*) \leftarrow \mathcal{A}^{\mathsf{Sim.Prove}(\mathsf{crs},\tau,\cdot)}(\mathsf{crs}) \ \wedge \ x^* \notin L \\ \qquad \wedge\ 1 \leftarrow \mathsf{Verify}(\mathsf{crs}, x^*, \pi^*) \mid (\mathsf{crs}, \tau) \leftarrow \mathsf{Sim.CRSGen}\left(1^k\right) \end{array}\right] = \mathsf{negl}(k)$$

where x^* is not in the list of queries made by $\mathcal{A}$ to Sim.Prove*.*

3.3 Commitment Schemes

A commitment scheme Com is a PPT algorithm that takes as input a string x and randomness r and outputs $c \leftarrow \mathsf{Com}(x; r)$. A perfectly binding commitment scheme must satisfy the following properties:

- **Perfectly Binding:** This property states that two different strings cannot have the same commitment. More formally, $\forall x_1 \neq x_2$ and r_1, r_2, $\mathsf{Com}(x_1; r_1) \neq \mathsf{Com}(x_2; r_2)$.
- **Computational Hiding:** For all strings x_0 and x_1 (of the same length), for all PPT adversaries $\mathcal{A}$, we have that:

$$|\Pr[A_1(\mathsf{Com}(x_0)) = 1] - Pr[A_1(\mathsf{Com}(x_1)) = 1)]| \leq \mathsf{negl}(k).$$

We note that it is in fact sufficient to use a standard 2-round statistically binding scheme in our construction in Section 4. Note that such a commitment scheme can be based on one way functions. For simplicitly of exposition, however, we will present our construction using a non-interactive perfectly binding scheme.

4 A Construction from Indistinguishability Obfuscation

Let $\mathcal{F}$ denote the family of all efficiently computable (deterministic) n-ary functions. We now present a functional encryption scheme $\mathcal{FE}_{\mathsf{I}}$ for $\mathcal{F}$. Assuming the existence of one-way functions and indistinguishability obfuscation for all efficiently computable circuits, we prove the following security guarantees for $\mathcal{FE}_{\mathsf{I}}$:

1. For $t = 0$, and any $q = q(k)$ such that $\binom{qn}{n} = \mathsf{poly}(k)$, $\mathcal{FE}_{\mathsf{I}}$ is $(0, q)$-SIM-secure.[2] In this case, the size of the secret keys in $\mathcal{FE}_{\mathsf{I}}$ grows linearly with $\binom{qn}{n}$.
2. For any $t \leq n$ and $q = \mathsf{poly}(k)$, $\mathcal{FE}_{\mathsf{I}}$ is (t, q)-sel-IND-secure. In this case, the size of the secret keys is independent of q.

Further, the size of each encryption key and ciphertext in $\mathcal{FE}_{\mathsf{I}}$ grows linearly with q. In Section 5, we give an efficient construction with "compact" encryption keys and ciphertexts, whose security is proven in the standard model.

Notation. Let $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ be a NIWI proof system. Let Com denote a perfectly binding commitment scheme. Let $i\mathcal{O}$ denote an indistinguishability obfuscator. Finally, let $\mathsf{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ be a semantically secure public-key encryption scheme. (See Section 3 for definitions of these notions.) We denote the length of ciphertexts in PKE by $\mathsf{c\text{-}len} = \mathsf{c\text{-}len}(k)$. Let $\mathsf{len} = 2 \cdot \mathsf{c\text{-}len}$.

We now proceed to describe our scheme $\mathcal{FE}_{\mathsf{I}} = (\mathsf{FE.Setup}, \mathsf{FE.Enc}, \mathsf{FE.Keygen}, \mathsf{FE.Dec})$.

Setup $\mathsf{FE.Setup}(1^k)$: The setup algorithm first computes a CRS $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^k)$ for the NIWI proof system. Next, it computes two key pairs – $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{PKE.Setup}(1^k)$ and $(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow \mathsf{PKE.Setup}(1^k)$ – of the public-key encryption scheme PKE. Finally, it computes the following commitments: (a) $Z_1^{i,j} \leftarrow \mathsf{Com}(0^{\mathsf{len}})$ for every $i \in [n]$, $j \in [q]$. (b) $Z_2^i \leftarrow \mathsf{Com}(0)$ for every $i \in [n]$.

For every $i \in [n]$, the i'th encryption key $\mathsf{EK}_i = \left(\mathsf{crs}, \mathsf{pk}_1, \mathsf{pk}_2, \left\{Z_1^{i,j}\right\}, Z_2^i, r_2^i\right)$ where r_2^i is the randomness used to compute the commitment Z_2^i. The master secret key is set to be $\mathsf{MSK} = \left(\mathsf{crs}, \mathsf{pk}_1, \mathsf{pk}_2, \mathsf{sk}_1, \left\{Z_1^{i,j}\right\}, \left\{Z_2^i\right\}\right)$. The setup algorithm outputs $(\mathsf{EK}_1, \ldots, \mathsf{EK}_n, \mathsf{MSK})$.

Encryption $\mathsf{FE.Enc}(\mathsf{EK}_i, x)$: To encrypt a message x with the i'th encryption key EK_i, the encryption algorithm first computes $c_1 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_1, x)$ and $c_2 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_2, x)$. Next, it computes a NIWI proof $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, y, w)$ for the statement $y = \left(c_1, c_2, \mathsf{pk}_1, \mathsf{pk}_2, \left\{Z_1^{i,j}\right\}, Z_2^i\right)$:

- *Either* c_1 and c_2 are encryptions of the same message and Z_2^i is a commitment to 0, *or*
- $\exists\ j \in [q]$ s.t. $Z_1^{i,j}$ is a commitment to $c_1 \| c_2$.

A witness $w_{\mathsf{real}} = (m, s_1, s_2, r_2^i)$ for the first part of the statement, referred to as the *real witness*, includes the message m and the randomness s_1 and s_2 used to compute the ciphertexts c_1 and c_2, respectively, and the randomness r_2^i used to compute Z_2^i. A witness $w_{\mathsf{trap}} = (j, r_1^{i,j})$ for the second part of the statement, referred to as the *trapdoor witness*, includes an index j and the randomness $r_1^{i,j}$ used to compute $Z_1^{i,j}$.

The honest encryption algorithm uses the real witness w_{real} to compute π. The output of the algorithm is the ciphertext $\mathsf{CT} = (c_1, c_2, \pi)$.

[2] Recall that when $t = 0$, there is no difference between selective security and standard security as defined in Section 2.2.2. See Remark 6.

Key Generation $\mathsf{FE.Keygen}(\mathsf{MSK}, f)$**:** The key generation algorithm on input f computes $\mathsf{SK}_f \leftarrow i\mathcal{O}(\mathsf{G}_f)$ where the function G_f is defined in Figure 1. Note that G_f has the master secret key MSK hardwired in its description.

$\mathsf{G}_f(\mathsf{CT}_1, \ldots, \mathsf{CT}_n)$

1. For every $i \in [n]$:
   (a) Parse $\mathsf{CT}_i = (c_{i,1}, c_{i,2}, \pi_i)$.
   (b) Let $y_i = \left(c_{i,1}, c_{i,2}, \mathsf{pk}_1, \mathsf{pk}_2, \left\{Z_1^{i,j}\right\}, Z_2^i\right)$ be the statement corresponding to the proof string π_i. If $\mathsf{Verify}(\mathsf{crs}, y_i, \pi_i) = 0$, then stop and output $\perp$. Otherwise, continue to the next step.
   (c) Compute $x_i \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}_1, c_{i,1})$.
2. Output $f(x_1, \ldots, x_n)$.

Figure 1: Functionality G_f

The algorithm outputs SK_f as the secret key for f.

Size of Function G_f*.* In order to prove that $\mathcal{FE}_{\mathsf{I}}$ is $(0, q)$-SIM-secure (see Section 4.2), we require the function G_f to be padded with zeros such that $|\mathsf{G}_f| = |\mathsf{Sim.G}_f|$, where the "simulated" functionality $\mathsf{Sim.G}_f$ is described later in Figure 2. In this case, the size of SK_f grows linearly with $\binom{qn}{n}$.

Note, however, that such a padding is *not* necessary to prove (t, q)-sel-IND-security for $\mathcal{FE}_{\mathsf{I}}$ (see Section 4.1). Indeed, in this case, the secret keys SK_f are independent of the number of message queries q made by the adversary.

Decryption $\mathsf{FE.Dec}(\mathsf{SK}_f, \mathsf{CT}_1, \ldots, \mathsf{CT}_n)$**:** The decryption algorithm on input $(\mathsf{CT}_1, \ldots, \mathsf{CT}_n)$ computes and outputs $\mathsf{SK}_f(\mathsf{CT}_1, \ldots, \mathsf{CT}_n)$.

This completes the description of the proposed functional encryption scheme $\mathcal{FE}_{\mathsf{I}}$. The correctness property of the scheme follows from inspection. We prove sel-IND security for $\mathcal{FE}_{\mathsf{I}}$ in Section 4.1, and then prove SIM security in Section 4.2.

4.1 Proving sel-IND Security

We now prove that the proposed scheme $\mathcal{FE}_{\mathsf{I}}$ is (t, q)-sel-IND-secure for any $t \leq n$.

Theorem 14. *Let $q = q(k)$ be a fixed $\mathsf{poly}(k)$. Then, assuming indistinguishability obfuscation for all polynomial-time computable circuits and one-way functions, the proposed scheme $\mathcal{FE}_{\mathsf{I}}$ is (t, q)-sel-IND-secure for any $t \leq n$.*

We prove the above theorem via a hybrid argument. We start by describing a sequence of hybrid experiments $\mathsf{H}_0, \ldots, \mathsf{H}_{10}$, where experiment H_0 (resp., H_{10}) corresponds to the real world experiment with challenge bit $b = 0$ (resp., $b = 1$). We will prove that for every i, the outputs of experiments H_i and H_{i+1} are computationally indistinguishable.

Hybrid H_0**:** This is the real experiment with challenge bit $b = 0$.

Hybrid H_1**:** This experiment is the same as H_0 except that the setup algorithm computes the commitments $\{Z_1^{i,j}\}$ in the following manner: let the challenge ciphertext $\hat{\mathsf{CT}}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$. Then, $Z_1^{i,j} \leftarrow \mathsf{Com}(\hat{c}_1^{i,j} \| \hat{c}_2^{i,j})$.

Hybrid H_2: This experiment is the same as H_1 except that in every challenge ciphertext $\hat{\mathsf{CT}}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$, the proof string $\hat{\pi}^{i,j}$ is computed using the *trapdoor* witness.

Hybrid H_3: This experiment is the same as H_2 except that for every $i \in \mathsf{N} \setminus \mathrm{I}$ (where I denotes the set of indices i s.t. EK_i is known to the adversary) the setup algorithm computes Z_2^i as a commitment to 1 (instead of 0). That is, for every $i \in [n]$, $Z_2^i \leftarrow \mathsf{Com}(1)$.

Hybrid H_4: This experiment is the same as H_3 except that in every challenge ciphertext $\hat{\mathsf{CT}}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$, the *second* ciphertext $\hat{c}_2^{i,j}$ is computed as an encryption of the challenge message $x_{i,j}^1$ (as opposed to $x_{i,j}^0$), i.e., $\hat{c}_2^{i,j} \leftarrow \mathsf{FE.Enc}(\mathsf{EK}_i, x_{i,j}^1)$.

Hybrid H_5: This experiment is the same as H_4 except that for every key query f, the corresponding secret key SK_f is computed as $\mathsf{SK}_f \leftarrow i\mathcal{O}(\mathsf{G}'_f)$ where G'_f is the same as the function G_f except that:

1. It has secret key sk_2 hardwired instead of sk_1.
2. It decrypts the *second* component of each input ciphertext using sk_2. More concretely, in step $1(c)$, plaintext x'_i is computed as $x'_i \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}_2, c_{i,2})$.

Hybrid H_6: This experiment is the same as H_5 except that in every challenge ciphertext $\hat{\mathsf{CT}}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$, the *first* ciphertext $\hat{c}_1^{i,j}$ is an encryption of challenge message $x_{i,j}^1$ (as opposed to $x_{i,j}^0$), i.e., $\hat{c}_1^{i,j} \leftarrow \mathsf{FE.Enc}(\mathsf{EK}_i, x_{i,j}^1)$.

Hybrid H_7: This experiment is the same as H_6 except that for every key query f, the corresponding secret key SK_f is computed as $\mathsf{SK}_f \leftarrow i\mathcal{O}(\mathsf{G}_f)$.

Hybrid H_8: This experiment is the same as H_7 except that the setup algorithm computes every Z_2^i as a commitment to 0, i.e., $Z_2^i \leftarrow \mathsf{Com}(0)$.

Hybrid H_9: This experiment is the same as H_8 except that in every challenge ciphertext $\hat{\mathsf{CT}}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$, the proof string $\hat{\pi}^{i,j}$ is computed using the *real* witness.

Hybrid H_{10}: This experiment is the same as H_9 except that the setup algorithm computes every $Z_1^{i,j}$ as a commitment to the all zeros string, i.e., $Z_1^{i,j} \leftarrow \mathsf{Com}(0^{\mathsf{len}})$. Note that this is the real experiment with challenge bit $b = 1$.

This completes the description of the hybrids. We argue their indistinguishability in Appendix A.

4.2 Proving SIM Security

Here we prove that the proposed scheme $\mathcal{FE}_{\mathsf{I}}$ is $(0, q)$-SIM-secure.

Theorem 15. *Let $q = q(k)$ be such that $\binom{qn}{n} = \mathsf{poly}(k)$. Then, assuming indistinguishability obfuscation for all polynomial-time computable circuits and one-way functions, the proposed scheme $\mathcal{FE}_{\mathsf{I}}$ is $(0, q)$-SIM-secure.*

In order to prove the above theorem, we first construct an ideal world adversary aka simulator $\mathcal{S}$. Then, in Appendix B, we prove indistinguishability of the outputs of the real and ideal world experiments via a hybrid argument.

Simulator $\mathcal{S}$. We describe a simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2)$ that only makes black-box use of a real-world adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$.

ALGORITHM $\mathcal{S}_0$. Let z be the auxiliary input given to $\mathcal{S}$. Algorithm $\mathcal{S}_0$ simply runs $\mathcal{A}_0$ with auxiliary input z and outputs $(\mathrm{I}, \mathsf{st}_0) \leftarrow \mathcal{A}_0(1^k, z)$. Since we are only considering the case where $t = 0$, we have that $\mathrm{I} = \emptyset$.

ALGORITHM $\mathcal{S}_1$. Algorithm $\mathcal{S}_1$ simply runs $\mathcal{A}_1$ on input st_0 and outputs $(\mathcal{M}, \mathsf{st}_1) \leftarrow \mathcal{A}_0(\mathsf{st}_0)$.

ALGORITHM $\mathcal{S}_2$. This algorithm runs the adversary algorithm $\mathcal{A}_2$ on simulated ciphertexts and provides simulated answers to the key queries made by $\mathcal{A}_2$. More concretely, $\mathcal{S}_2$ runs in the following sequence of steps:

1. **Simulate Setup.** $\mathcal{S}_2$ first performs a simulated setup procedure. Namely, it first computes a CRS $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^k)$ for the NIWI proof system. Next, it computes two key pairs – $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{PKE.Setup}(1^k)$ and $(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow \mathsf{PKE.Setup}(1^k)$ – of the public-key encryption scheme PKE. Finally, it computes the commitments $\{Z_1^{i,j}\}$ and $\{Z_2^i\}$ in the following manner:
   - For every $i \in [n]$, $j \in [q]$: (a) Compute $\hat{c}_1^{i,j}$ and $\hat{c}_2^{i,j}$ as encryptions of zeros, i.e., $\hat{c}_1^{i,j} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_1, 0)$ and $\hat{c}_2^{i,j} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_2, 0)$. (b) Compute $Z_1^{i,j} \leftarrow \mathsf{Com}(\hat{c}_1^{i,j} \| \hat{c}_2^{i,j})$. Let $r_1^{i,j}$ be the randomness used to compute $Z_1^{i,j}$..
   - For every $i \in [n]$, compute Z_2^i as a commitment to 1, i.e., $Z_2^i \leftarrow \mathsf{Com}(1)$.

   Let $\mathsf{MSK} = \left(\mathsf{crs}, \mathsf{pk}_1, \mathsf{pk}_2, \mathsf{sk}_1, \left\{Z_1^{i,j}\right\}, \left\{Z_2^i\right\}\right)$.
2. **Simulate Challenge Ciphertexts.** $\mathcal{S}_2$ now computes simulated challenge ciphertexts $\vec{\mathsf{CT}} = \{\hat{\mathsf{CT}}_{1,j}, \ldots, \hat{\mathsf{CT}}_{n,j}\}_{j=1}^q$ in the following manner. For every $i \in [n]$, $j \in [q]$:
   - Let $y_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \mathsf{pk}_1, \mathsf{pk}_2, \{Z_1^{i,j}\}, Z_2^i)$. Compute the proof $\hat{\pi}_i \leftarrow \mathsf{Prove}(\mathsf{crs}, y_{i,j}, w_{i,j})$ where the witness $w_{i,j}$ corresponds to the *trapdoor witness* $(j, r_1^{i,j})$. That is, $w_{i,j}$ establishes that $Z_1^{i,j}$ is a commitment to $\hat{c}_1^{i,j} \| \hat{c}_2^{i,j}$.
   - The simulated ciphertext $\hat{\mathsf{CT}}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$.
3. **Simulate Key Queries.** Finally, $\mathcal{S}_2$ runs the adversary algorithm $\mathcal{A}_2$ on input $(\vec{\mathsf{CT}}, \mathsf{st}_1)$. Recall from Definition 4 that $\mathcal{A}_2$ also makes queries to the key generation oracle. $\mathcal{S}_2$ simulates responses to $\mathcal{A}_2$'s key queries in the following manner. Let TP denote the ideal world trusted party that given the message distribution $\mathcal{M}$ (output by $\mathcal{S}_2$) first samples $\vec{X} \leftarrow M$, where $\vec{X} = \{x_{1,j}, \ldots, x_{n,j}\}_{j=1}^q$. When $\mathcal{A}_2$ makes a key query f, $\mathcal{S}_2$ performs the following sequence of steps:
   - Query the trusted party TP on function $(f, j_1, \ldots, j_n)$ for every choice of $j_1, \ldots, j_n \in [q]$. The trusted party computes and returns the function outputs $\mathsf{out}[j_1, \ldots, j_n] = f(x_{1,j_1}, \ldots, x_{n,j_n}\}$ to $\mathcal{S}_2$. Let $\vec{\mathsf{out}}$ denote the vector of all the $\binom{q}{n}$ number of outputs.
   - Compute the secret key SK_f for function f as $\mathsf{SK}_f \leftarrow i\mathcal{O}(\mathsf{Sim.G}_f)$. The functionality $\mathsf{Sim.G}_f$ has the master secret key MSK, the challenge ciphertext pairs $\{\hat{c}_1^{i,j}, \hat{c}_2^{i,j}\}$ and the outputs $\vec{\mathsf{out}}$ hardwired in it. It is described in Figure 2.
   - Return SK_f to $\mathcal{A}_2$.

Finally, at some point $\mathcal{A}_2$ outputs its view α. $\mathcal{S}_2$ outputs α and stops.

In Appendix B, we prove indistinguishability of the outputs of the real and ideal experiments.

5 A Construction from Differing-Inputs Obfuscation

Let $\mathcal{F}$ denote the family of all efficiently computable (deterministic) n-ary functions. We now present a new functional encryption scheme $\mathcal{FE}_{\mathsf{II}}$ for $\mathcal{F}$ based on differing-inputs obfuscation.

$\mathsf{Sim.G}_f(\mathsf{CT}_1, \ldots, \mathsf{CT}_n)$

1. For every $i \in [n]$:
   - Parse $\mathsf{CT}_i = (c_{i,1}, c_{i,2}, \pi_i)$.
   - Let $y_i = \left(c_{i,1}, c_{i,2}, \mathsf{pk}_1, \mathsf{pk}_2, \left\{Z_1^{i,j}\right\}, Z_2^i\right)$ be the statement corresponding to the proof string π_i. If $\mathsf{Verify}(\mathsf{crs}, y_i, \pi_i) = 0$, then stop and output $\perp$. Otherwise, continue to the next step.
2. If $\exists\ (j_1, \ldots, j_n)$ s.t. for every $i \in [n]$,
   - $\hat{c}_1^{i,j_i} = c_{i,1}$, and
   - $\hat{c}_2^{i,j_i} = c_{i,2}$,

   then stop and output $\mathsf{out}[j_1, \ldots, j_n]$.
3. Otherwise, for every $i \in [n]$,
   - Compute $x_i \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}_1, c_{i,1})$.
4. Output $f(x_1, \ldots, x_n)$.

Figure 2: Functionality $\mathsf{Sim.G}_f$

The main advantage of this scheme over the one presented in Section 4 is that the encryption keys and the ciphertexts are "compact", i.e., independent of the number of message queries q made by the adversary.

The proposed scheme provides the following security guarantees:

- For any choice of $t \leq n$, $\mathcal{FE}_{\mathsf{II}}$ is $(t, \mathsf{poly}(k))$-IND-secure. In this case, the number of message queries q can be an arbitrary unbounded polynomial $q = \mathsf{poly}(k)$.
- For $t = 0$ and $q = q(k)$ such that $\binom{qn}{n} = \mathsf{poly}(k)$, our construction naturally extends to $(0, q)$-SIM-security. In this case, the size of the secret keys grows linearly with $\binom{qn}{n}$.

Notation. Let $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ be a simulation-sound NIZK argument system. Let Com denote a perfectly binding commitment scheme. Let $\mathsf{di}\mathcal{O}$ denote a differing-inputs obfuscator. Finally, let $\mathsf{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ be a semantically secure public-key encryption scheme.

We now proceed to describe the scheme $\mathcal{FE}_{\mathsf{II}} = (\mathsf{FE.Setup}, \mathsf{FE.Enc}, \mathsf{FE.Keygen}, \mathsf{FE.Dec})$.

Setup $\mathsf{FE.Setup}(1^k)$: The setup algorithm first computes a CRS $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^k)$ for the simulation-sound NIZK proof system. Next, it computes two key pairs – $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{PKE.Setup}(1^k)$ and $(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow \mathsf{PKE.Setup}(1^k)$ – of the public-key encryption scheme PKE. Finally, for every $i \in [n]$, it computes a commitment $Z_i \leftarrow \mathsf{Com}(0)$.

For every $i \in [n]$, the i'th encryption key $\mathsf{EK}_i = (\mathsf{crs}, \mathsf{pk}_1, \mathsf{pk}_2, Z_i, r_i)$ where r_i is the randomness used to compute Z_i. The master secret key $\mathsf{MSK} = (\mathsf{crs}, \mathsf{pk}_1, \mathsf{pk}_2, \mathsf{sk}_1, \{Z_i\})$. The setup algorithm outputs $(\mathsf{EK}_1, \ldots, \mathsf{EK}_n, \mathsf{MSK})$.

Encryption $\mathsf{FE.Enc}(\mathsf{EK}_i, x)$: To encrypt a message x with the i'th encryption key EK_i, the encryption algorithm first computes $c_1 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_1, x)$ and $c_2 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_2, x)$. Next, it computes a simulation-sound NIZK proof $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, y, w)$ for the statement $y = (c_1, c_2, \mathsf{pk}_1, \mathsf{pk}_2, Z)$:

- c_1 and c_2 are encryptions of the same message *and* Z_i is a commitment to 0.

Here, a witness $w = (s_1, s_2, r_i)$ for y consists of the randomness s_1 and s_2 used to compute c_1 and c_2, respectively, and the randomness r_i used to compute Z_i.

The output of the algorithm is the ciphertext $\mathsf{CT} = (c_1, c_2, \pi)$.

Key Generation $\mathsf{FE.Keygen}(\mathsf{MSK}, f)$**:** The key generation algorithm on input f computes $\mathsf{SK}_f \leftarrow \mathsf{di}\mathcal{O}(\mathcal{H}_f)$ where the function $\mathcal{H}_f$ is defined in Figure 3. Note that $\mathcal{H}_f$ has the master secret key MSK hardwired in its description.

$\mathcal{H}_f(\mathsf{CT}_1, \ldots, \mathsf{CT}_n)$

1. For every $i \in [n]$:
   (a) Parse $\mathsf{CT}_i = (c_{i,1}, c_{i,2}, \pi_i)$.
   (b) Let $y_i = (c_{i,1}, c_{i,2}, \mathsf{pk}_1, \mathsf{pk}_2, \{Z_i\})$ be the statement corresponding to the proof string π_i. If $\mathsf{Verify}(\mathsf{crs}, y_i, \pi_i) = 0$, then stop and output $\perp$. Otherwise, continue to the next step.
   (c) Compute $x_i \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}_1, c_{i,1})$.
2. Output $f(x_1, \ldots, x_n)$.

Figure 3: Functionality $\mathcal{H}_f$

The algorithm outputs SK_f as the secret key for f.

Size of Function $\mathcal{H}_f$*.* Similar to the construction in Section 4, in order to prove that $\mathcal{FE}_{\mathsf{II}}$ is $(0, q)$-SIM-secure, we require the function $\mathcal{H}_f$ to be padded with zeros such that the size of $\mathcal{H}_f$ is equal to the size of its "simulated version" which has (among other things) $\binom{qn}{n}$ output values hardwired in it. Thus, in this case, the size of SK_f grows linearly with $\binom{qn}{n}$.

Note, however, that such a padding is *not* necessary to prove (t, q)-sel-IND-security for $\mathcal{FE}_{\mathsf{II}}$. Indeed, in this case, the secret keys SK_f are independent of the number of message queries q made by the adversary.

Decryption $\mathsf{FE.Dec}(\mathsf{SK}_f, \mathsf{CT}_1, \ldots, \mathsf{CT}_n)$**:** The decryption algorithm on input $(\mathsf{CT}_1, \ldots, \mathsf{CT}_n)$ computes and outputs $\mathsf{SK}_f(\mathsf{CT}_1, \ldots, \mathsf{CT}_n)$.

This completes the description of $\mathcal{FE}_{\mathsf{II}}$. The correctness property of the scheme follows from inspection.

Theorem 16. *Assuming differing-inputs obfuscation for all polynomial-time computable circuits and one-way functions, the proposed scheme* $\mathcal{FE}_{\mathsf{II}}$ *is* $(t, \mathsf{poly}(k))$*-*IND*-secure for any* $t \leq n$*.*

We prove the above theorem in Appendix C. Further, we note that for $t = 0$ and $q = q(k)$ such that $\binom{qn}{n} = \mathsf{poly}(k)$, our IND-security proof can be naturally extended to argue $(0, q)$-SIM-security for $\mathcal{FE}_{\mathsf{II}}$ by using a similar simulation strategy as for our first construction (see Section 4). We formally state the claim below, but omit the proof details from this manuscript.

Theorem 17. *Let* $q = q(k)$ *be such that* $\binom{qn}{n} = \mathsf{poly}(k)$*. Then, assuming differing-inputs obfuscation for all polynomial-time computable circuits and one-way functions, the proposed scheme* $\mathcal{FE}_{\mathsf{II}}$ *is* $(0, q)$*-*SIM*-secure.*

6 Multi-Input Functional Encryption Implies Obfuscation

In this section, we prove that various flavors of multi-input FE imply well established notions of program obfuscation.

Indistinguishability Obfuscation from MI-FE. Our first result shows that the indistinguishability notion of multi-input FE unconditionally implies indistinguishability obfuscation (note that such an implication is not known to hold for single input FE). This, in particular, means that the use of indistinguishability obfuscation is unavoidable for multi-input FE, and, any future improvements in the complexity assumptions on which multi-input FE is based will only come with a corresponding improvement in the indistinguishability obfuscation constructions. We state the theorem below for the "weakest" case of *secret-key* multi-input functional encryption (this only strengthens our result).

Theorem 18. $(0,2)$-IND*-secure MI-FE for general* $(k+1)$*-ary functions unconditionally implies indistinguishability obfuscation for all circuits with* k*-bit inputs.*

Proof. We describe how to construct indistinguishability obfuscation for a circuit class $\mathcal{C}$ where for every $C \in \mathcal{C}$, $C : \{0,1\}^k \to \{0,1\}^{k'}$ and $|C| = \ell$. Let $\mathcal{FE}$ be a $(0,2)$-IND-secure MI-FE scheme for general $(k+1)$-ary functions. The PPT obfuscator $i\mathcal{O}$ works as follows.

- Consider a function g s.t. $g(x_1, \ldots, x_k, C) = C(x_1 || \ldots || x_k)$ where for all i, $x_i \in \{0,1\}$, and, $C \in \{0,1\}^\ell$. Observe that the function g acts as a universal circuit and treats its $(k+1)$-th input as a circuit.
- The obfuscator $i\mathcal{O}$ first runs the setup algorithm for $\mathcal{FE}$ to compute a master secret key MSK and encryption keys as $\mathsf{EK}_1, \ldots, \mathsf{EK}_{k+1}$. It then runs the key generation algorithm of $\mathcal{FE}$ to generate a secret key for the function g using MSK. Denote the resulting decryption key as SK_g.
- For all $i \in [k], b \in \{0,1\}$, let $\mathsf{CT}_i^b \leftarrow \mathsf{FE.Enc}(\mathsf{EK}_i, b)$. All the encryptions are performed using independent random coins. Furthermore, let $\mathsf{CT}_{k+1} \leftarrow \mathsf{FE.Enc}(\mathsf{EK}_{k+1}, C)$.
- The obfuscated circuit $i\mathcal{O}(C) = (\{\mathsf{CT}_i^b\}_{b,i}, \mathsf{CT}_{k+1}, \mathsf{SK}_g)$.

To evaluate the obfuscated circuit on an input $x = (x_1, \ldots, x_k)$, simply evaluate the decryption algorithm $\mathsf{FE.Dec}(\mathsf{SK}_g, \{\mathsf{CT}_i^{x_i}\}_i, \mathsf{CT}_{k+1})$. This results in $g(x_1, \ldots, x_k, C) = C(x)$. This completes the description of the obfuscation scheme.

We now show that the above construction is indeed a secure indistinguishability obfuscation. This follows from the $(0,2)$-IND-security of the underlying multi-input FE scheme. Consider any two functionally equivalent circuit C_0 and C_1 from $\mathcal{C}$. That is, for all $x \in \{0,1\}^k, C_0(x) = C_1(x)$. Now, suppose for contradiction that there exists a PPT adversary $\mathcal{A}$ that distinguishes between $i\mathcal{O}(C_0)$ and $i\mathcal{O}(C_1)$ with non-negligible advantage. We will construct an adversary $\mathcal{B}$ that breaks $(0,2)$-IND-security of $\mathcal{FE}$. The adversary $\mathcal{B}$ runs $\mathcal{A}$ and receives circuits C_0 and C_1. It works as follows:

1. It defines challenge message vectors $\vec{X}^0$ and $\vec{X}^1$, where $\vec{X}^b = \{x_{1,j}^b, \ldots, x_{k+1,j}^b\}_{j\in[2]}$. For every $i \in [k]$, set $x_{i,1}^b = 0$ and $x_{i,2}^b = 1$. (Note here that $x_{i,j}^0 = x_{i,j}^1$.) Further, set $x_{k+1,1}^b = x_{k+1,2}^b = C_b$. $\mathcal{B}$ sends over $\vec{X}^0$, $\vec{X}^1$ to the challenger in the IND security game. Let $\{\mathsf{CT}_{1,j}, \ldots, \mathsf{CT}_{k+1,j}\}_{j\in[2]}$ denote the challenge ciphertexts received by $\mathcal{B}$
2. Next, $\mathcal{B}$ requests a secret key for the function g. Let SK_g be the secret key received by $\mathcal{B}$.
3. Now, $\mathcal{B}$ sends over $\left(\{\mathsf{CT}_{1,j}, \ldots, \mathsf{CT}_{k,j}\}_{j\in[2]}, \mathsf{CT}_{k+1,1}\right)$ as the challenge obfuscation to $\mathcal{A}$. $\mathcal{B}$ simply outputs the guess b' returned by $\mathcal{A}$.

This completes the description of $\mathcal{B}$. We first argue that the challenge message vectors and the secret key query g are I-compatible as per IND security definition 3. (Here $\mathrm{I} = \emptyset$.) To see this, note that for any $x = (x_1, \ldots, x_k)$, we have that:

$$g(x_1, \ldots, x_k, C_0) = g(x_1, \ldots, x_k, C_1).$$

since C_0 and C_1 are functionally equivalent. Now, note that if the challenge bit b chosen by IND-security game challenge is equal to 0, then the resulting obfuscation is sent by $\mathcal{B}$ to $\mathcal{A}$ is of circuit C_0; otherwise it is an obfuscation of C_1. Thus, if $\mathcal{A}$ can distinguish between these two cases with non-negligible advantage, then $\mathcal{B}$ also wins the IND game with the same advantage. This completes the proof. □

Remark 19. *We remark that in the above proof, the "order" of the key query g is irrelevant. That is, g could be queried* before *or* after *the ciphertext queries.*

Virtual Black-Box Obfuscation from MI-FE. We now give two results for constructing virtual black-box obfuscation from various flavors of MI-FE. We first note that the same construction as above (Theorem 18), in fact, implies virtual black-box obfuscation, when $\mathcal{FE}$ is $(0, 2)$-SIM-secure.

Theorem 20. $(0, 2)$-SIM*-secure FE for general $(k + 1)$-ary functions unconditionally implies virtual black-box obfuscation for all circuits with k-bit inputs.*

Next, we show that SIM-secure multi-input FE, where at least one of the encryption keys may be made public, implies virtual black-box obfuscation.

Theorem 21. $(1, 1)$-SIM*-secure MI-FE for general 2-ary functions unconditionally implies virtual black-box obfuscation for all circuits.*

Sketch. The proof of this theorem is quite similar to that of the previous one and we only provide a sketch here. The basic idea, as before, is to give out keys for a universal circuit g. The first input to g will be the function f which we wish to obfuscate. The encryption key EK_1 will be kept a secret, and, the ciphertext $\mathsf{FE.Enc}(\mathsf{EK}_1, f)$ will be included as part of the obfuscated circuit. The second input x will be the input on which the user wishes to evaluate f. Hence, the user is given access to the second encryption key EK_2 (as part of the obfuscated circuit) to enable it to encrypt any x. More details follow:

- Consider a function g s.t. $g(C, x) = C(x)$. Let $\mathcal{FE}$ be a $(1, 1)$-SIM-secure MI-FE for general 2-ary functions. The obfuscator VBB runs the setup algorithm for $\mathcal{FE}$ to compute MSK and encryption keys $(\mathsf{EK}_1, \mathsf{EK}_2)$. It then runs the key generation algorithm of $\mathcal{FE}$ to generate a secret key SK_g for the above function g using MSK.
- Let $\mathsf{CT} \leftarrow \mathsf{FE.Enc}(\mathsf{EK}_1, C)$. The obfuscated circuit $\mathsf{VBB}(C) = (\mathsf{CT}, \mathsf{EK}_2, \mathsf{SK}_g)$.

To evaluate the obfuscated circuit on an input x, compute $\mathsf{CT}' \leftarrow \mathsf{FE.Enc}(\mathsf{EK}_2, x)$, and, run $\mathsf{FE.Dec}(\mathsf{SK}_g, \mathsf{CT}, \mathsf{CT}')$. This results in $g(C, x) = C(x)$.

The virtual black-box obfuscation property follows from the fact that the view of the user can be simulated given access to a trusted party holding the first input f, and, evaluating $g(f, \cdot)$ on any second input x of user's choice. □

6.1 Impossibility Results for SIM secure MI-FE

Here, we discuss some impossibility results for simulation secure MI-FE that complement our positive results given in Sections 4 and 5.

Recall that [BSW11, BO13] already establish the impossibility of $(0, \mathsf{poly}(k))$-SIM-secure functional encryption for 1-ary functions. We show that for n-ary functions, where $n \geq 2$, the

situation is much worse. In particular, recall that Barak et al. [BGI+01] proved an (unconditional) impossibility result for VBB obfuscation for general circuits. Then, combining their result with Theorem 21, we get the following result:

Theorem 22. $(1,1)$-SIM-*secure multi-input functional encryption for general* 2-*ary functions is impossible.*

We remark that our positive results for SIM-secure MI-FE in Sections 4 and 5 are consistent with the above negative result and that of [BSW11, BO13].

Simulation secure MI-FE against Non-Adaptive Key Queries. So far in this paper, we have only considered simulation security for MI-FE in the setting where an adversary makes key queries *after* choosing the challenge messages. Following the terminology from the literature on single-input functional encryption, such queries are referred to as *adaptive* key queries. One can consider the "opposite" scenario, where the adversary is allowed to make key queries *before* choosing the challenge messages. This setting has been well studied in the case of single-input functional encryption, where such queries are referred to as *non-adaptive* key queries.

We now discuss the feasibility of simulation-based security for non-adaptive key queries (referred to as NA-SIM security) in our setting of multi-input FE. NA-SIM security for multi-input FE is defined similarly to definition 4, except that now the adversary is required to make key queries *before* (as opposed to after) choosing the challenge messages. More concretely, we can define (t,p,q)-NA-SIM-secure functional encryption where (as earlier) t denotes the number of encryption keys known to the adversary and q denotes the number of challenge messages per encryption key. The new parameter p denotes the total number of non-adaptive key queries by the adversary. For completeness, we provide a formal definition in Appendix D.

Now, observe that the proofs of Theorem 21 and Theorem 20 are insensitive to the "order" of the key query; i.e., they go through even if the key query is *non-adaptive*. Then, combining these results with the impossibility result of Barak et al [BGI+01], we obtain the following two (incomparable) results:

Theorem 23. $(1,1,1)$-NA-SIM-*secure multi-input functional encryption for general* 2-*ary functions is impossible.*

Theorem 24. $(0,1,2)$-NA-SIM-*secure multi-input functional encryption for general* $(k+1)$-*ary functions is impossible.*

We remark that we have stated Theorem 24 for the secret-key setting (as opposed to for general t) since it is the "weakest" case, and therefore only strengthens our result.

While the above impossibility results rule out achieving NA-SIM-security for general functions – in particular, they rule out NA-SIM-security for the arguably unnatural function that cannot be VBB obfuscated [BGI+01]) – we also provide another impossibility result for the weak pseudo-random function.

Let $\{F\}$ be a weak pseudo-random function family with key space K and message space X. The 2-ary $\mathsf{wPRF}(\cdot,\cdot)$ functionality on input key $k \in K$ and message $x \in X$ outputs $F_K(x)$. We shall call k as the *first* input and x to be the *second* input to wPRF. We claim the following:

Theorem 25. $(0,1,\mathsf{poly}(k))$-NA-SIM-*secure functional encryption for the weak PRF functionality* $\mathsf{wPRF}(\cdot,\cdot)$ *is impossible.*

Proof. (Sketch). Here, we sketch a proof for black-box simulation. The proof follows along the same lines as in [AGVW13]. Suppose for contradiction that there exists a $(0,1,\mathsf{poly}(k))$-NA-SIM-secure functional encryption $\mathcal{FE}$ for the weak PRF functionality. Let $\ell - 1$ denote an upper bound on the ciphertext size in $\mathcal{FE}$. We construct an adversary $\mathcal{A}$ that makes a single

key query and ℓ^2 number of message queries (per encryption key) such that every (black-box) simulator "fails" to simulate the view of $\mathcal{A}$.

The adversary $\mathcal{A}$ first makes a single (non-adaptive) key query for the 2-ary function wPRF. Let $L = \ell^2$. Then, $\mathcal{A}$ asks ciphertexts for L first inputs $k_1, \ldots, k_L$ and L second inputs $x_1, \ldots, x_L$, where each k_i is chosen uniformly at random from the key space K and each x_i is chosen uniformly at random from the message space X. Now the simulator first needs to produce a key $\mathsf{SK}_{\mathsf{wPRF}}$ and then it is given the functionality's outputs $\{\mathsf{wPRF}(k_i, x_j)\}_{i=1,j=1}^{L,L}$. Now, the simulator has to produce $2L$ ciphertexts $\left\{\mathsf{CT}_i^1\right\}_{i=1}^{L}$, $\left\{\mathsf{CT}_j^2\right\}_{j=1}^{L}$ such that for every $i \in [L]$, $j \in [L]$, $\mathsf{wPRF}(k_i, x_j) = \mathsf{FE.Dec}(\mathsf{SK}_f, \mathsf{CT}_i^1, \mathsf{CT}_j^2)$.

Thus, on the one hand, the simulator needs to "encode" all of the functionality's outputs into $2L$ ciphertexts. On the other hand, the functionality's outputs are $L^2 = \ell^4$ pseudo-random bits, while the total length of the $2L$ ciphertexts is $2L(\ell - 1) < 2\ell^3$ bits. Since a pseudo-random string cannot be efficiently compressed, we get a contradiction. □

Discussion. Recall that the lower bounds of [AGVW13, CIJ+13] already establish that it is impossible to achieve $(0, \mathsf{poly}(k), 1)$-$\mathsf{NA\text{-}SIM}$-secure functional encryption for 1-ary functions (specifically, the weak PRF functionality). That is, it is impossible to achieve $\mathsf{NA\text{-}SIM}$ security against an unbounded number of non-adaptive key queries even in the secret-key setting. Our impossibility results in Theorem 24 and Theorem 25 establish that it is also impossible to achieve $\mathsf{NA\text{-}SIM}$ security against an unbounded number of ciphertext queries. Thus, $\mathsf{NA\text{-}SIM}$ secure MI-FE is only possible for a *bounded* number of key queries and a *bounded* number of ciphertext queries. This is strictly worse that what can be achieved in the case of SIM security (where *unbounded* number of key queries can be achieved, in the secret-key setting, as exemplified by our positive results).

7 Extension to Randomized Functionalities

Our positive results for multi-input functional encryption presented in Sections 4 and 5 only concern with deterministic n-ary functions. Here, we discuss how to extend our results to handle *randomized* functionalities.

Modeling Security. In the single-input setting, the case of randomized functionalities was recently considered by Goyal et al. [GJKS13]. Very briefly, Goyal et al. observed that in the setting of randomized functionalities, the central challenge is to ensure that the random coins used for computing a function output are unbiased and remain hidden from the participants (i.e., the encryptor/sender and the decryptor/receiver). As such, in addition to requiring security against dishonest receivers, one must explicitly require security against dishonest senders to ensure that it is not possible to force "bad" outputs on an honest receiver.

We follow the same approach in our multi-input setting. Specifically, following [GJKS13], below we formalize a definition for security against dishonest senders. Overall, we will say that a multi-input functional encryption scheme for a randomized function family is secure if it achieves security against both dishonest senders and dishonest receivers.

Definition 26 (Security against Dishonest Senders)**.** *We say that a functional encryption scheme $\mathcal{FE}$ for n-ary (randomized) functions $\mathcal{F}$ is* t-secure against dishonest senders *if for every PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$, there exists a PPT simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2)$ such that the outputs of the following two experiments are computationally indistinguishable:*

Experiment $\mathsf{REAL}_{\mathcal{A}}^{\mathcal{FE}}(1^k)$:
$(\mathrm{I}, \mathsf{st}_0) \leftarrow \mathcal{A}_0(1^k)$ *where* $|\mathrm{I}| = t$
$(\vec{\mathsf{EK}}, \mathsf{MSK}) \leftarrow \mathsf{FE.Setup}(1^k)$
$(\{f\}, \mathcal{M}, \mathsf{st}_1) \leftarrow \mathcal{A}_1(\mathsf{st}_0, \vec{\mathsf{EK}}_{\mathrm{I}})$
$\mathsf{SK}_f \leftarrow \mathsf{FE.Keygen}(\mathsf{MSK}, f)\ \forall\ f \in \{f\}$
$\mathsf{CT}_i \leftarrow \mathsf{FE.Enc}(\mathsf{EK}_i, x_i)$ *where* $x_i \leftarrow \mathcal{M}\ \forall\ i \in \mathsf{N} \setminus \mathrm{I}$
$\alpha \leftarrow \mathcal{A}_2^{\mathcal{O}\left(\{\mathsf{SK}_f\}, \vec{\mathsf{CT}}, \cdot\right)}(\vec{\mathsf{CT}}, \mathsf{st}_1)$
Output: $(\mathrm{I}, \{f\}, \mathcal{M}, \vec{x}, \{\mathsf{out}\}, \alpha)$

Experiment $\mathsf{IDEAL}_{\mathcal{S}}^{\mathcal{FE}}(1^k)$:
$(\mathrm{I}, \mathsf{st}_0) \leftarrow \mathcal{S}_0(1^k)$
$(\{f\}, \mathcal{M}, \mathsf{st}_1) \leftarrow \mathcal{S}_1(\mathsf{st}_0)$
$x_i \leftarrow \mathcal{M}\ \forall\ i \in \mathsf{N} \setminus \mathrm{I}$
$\alpha \leftarrow \mathcal{S}_2^{\mathsf{TP}(\{f\}, \vec{x}, \cdot)}(\mathsf{st}_1)$
Output: $(\mathrm{I}, \{f\}, \mathcal{M}, \vec{x}, \{\mathsf{out}'\}, \alpha)$

where,

- *In the real world, oracle* $\mathcal{O}\left(\{\mathsf{SK}_f\}, \vec{\mathsf{CT}}, \cdot\right)$ *accepts queries of the form* $(\mathsf{CT}_1^*, \ldots, \mathsf{CT}_t^*)$ *such that for every* $i \in [t]$, $j \in \mathsf{N} \setminus \mathrm{I}$, $\mathsf{CT}_i^* \neq \mathsf{CT}_j$. *It outputs* $\mathsf{FE.Dec}\left(\mathsf{SK}_f, \langle \mathsf{CT}_1, \ldots, \mathsf{CT}_{n-t}, \mathsf{CT}_1^*, \ldots, \mathsf{CT}_t^* \rangle\right)$ *for every* $\mathsf{SK}_f \in \{\mathsf{SK}_f\}$. *Here,* $\langle z_{i_1}, \ldots, z_{i_n} \rangle$ *denotes a permutation of the ciphertexts* $z_{i_1}, \ldots, z_{i_n}$ *such that* z_{i_j} *is mapped to the* ℓ*'th location if* z_{i_j} *is the encrypted via the encryption key* EK_ℓ. *Further,* $\{\mathsf{out}\}$ *denotes the set of outputs of* $\mathcal{O}$ *to* $\mathcal{A}_2$*'s decryption queries.*
- *In the ideal world, the trusted party* $\mathsf{TP}\left(\{f\}, \vec{x}, \cdot\right)$ *accepts queries of the form* $(x_1^*, \ldots, x_t^*)$ *and outputs* $f_j\left(\left\langle x_{i_1}, \ldots, x_{i_{n-t}}, x_1^*, \ldots, x_t^* \right\rangle; r_j\right)$ *for every* $f_j \in \{f\}$. *Here* r_j *is chosen uniformly at random and* $\langle z_{i_1}, \ldots, z_{i_n} \rangle$ *denotes a permutation of the values* $z_{i_1}, \ldots, z_{i_n}$ *such that the value* z_{i_j} *is mapped to the* ℓ*'th location if* z_{i_j} *is the* ℓ*'th input (out of* n *inputs) to* f. *Further,* $\{\mathsf{out}'\}$ *denotes the set of outputs of* TP *to the queries of* $\mathcal{S}_2$.

We now define SIM security for multi-input functional encryption for randomized functions. We note that IND security can be defined analogously; we skip the details.

Definition 27. *We say that a functional encryption scheme* $\mathcal{FE}$ *for* n*-ary (randomized) functions* $\mathcal{F}$ *is* (t_1, t_2, q)-SIM*-secure if:*

1. $\mathcal{FE}$ *is* t_1*-secure against dishonest senders.*
2. $\mathcal{FE}$ *is* (t_2, q)-SIM*-secure against dishonest receivers.*

Positive Results for Randomized Functionalities. Building on the techniques of [GJKS13], both of our constructions for multi-input functional encryption presented in Sections 4 and 5 can be extended to handle randomized functionalities. Below, we outline the necessary modifications to our second scheme $\mathcal{FE}_{\mathsf{II}}$ to define a new scheme $\mathcal{FE}$. (We note that $\mathcal{FE}_{\mathsf{I}}$ can be modified in a similar manner to handle randomized functionalities.)

$\mathcal{FE}$ is defined similarly to $\mathcal{FE}_{\mathsf{II}}$, with the following necessary changes:

1. To encrypt a message x, we follow the same steps as in $\mathcal{FE}_{\mathsf{II}}$ to compute (c_1, c_2, π). Next, we sample a key pair (sk, vk) for a strongly unforgeable one-time signature scheme. The final ciphertext CT consists of $(c_1, c_2, \pi, vk, \sigma)$, where σ is a signature over $c_1 \| c_2 \| \pi$ using sk.
2. To compute a secret key SK_f for a (randomized) function f, we first sample a key K for a *puncturable* pseudo-random function (PRF) [SW13, BW13, BGI13, KPTZ13]. Then, key SK_f is computed as $\mathsf{di}\mathcal{O}(\mathcal{H}'_f)$ where $\mathcal{H}'_f$ is defined similarly to the functionality $\mathcal{H}_f$ except that:
   - We additionally check whether the signature σ_i in each input ciphertext CT_i is valid.
   - Further, after decrypting each input ciphertext CT_i to compute x_i, we first compute randomness r as the output of the PRF on input $\mathsf{CT}_1 \| \ldots \| \mathsf{CT}_n$ using key K. The final output is then computed as $f(x_1, \ldots, x_n; r)$.

Very briefly, security against dishonest senders follows from the same ideas as in [NY90, DDN91, Sah99]. Specifically, incorporating the one-time signatures in the ciphertexts ensures that each ciphertext is unique (and therefore, an adversary cannot modify an honest sender's ciphertext to create a decryption query). Further, it is possible to extract the input from an adversarially created ciphertext using one of the secret keys (while using the semantic security for the other key). Security against dishonest receivers follows largely in the same manner as for $\mathcal{FE}_{\mathsf{II}}$. The main difference now is that (as in [GJKS13]), we use the punctured PRF to remove all the secret information in the PRF key for the point $\hat{\mathsf{CT}}_1 \| \ldots \| \hat{\mathsf{CT}}_n$, where $\hat{\mathsf{CT}}_1, \ldots, \hat{\mathsf{CT}}_n$ denotes a challenge ciphertext tuple. From the security of the obfuscation, it follows that this randomness remains hidden from a honest receiver. We refer the reader to [GJKS13] for more details on the proof.

References

[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.

[AGVW13] Shweta Agrawal, Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption: New perspectives and lower bounds. In *CRYPTO (2)*, 2013.

[BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *ITCS*, pages 326–349, 2012.

[BCLO09] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'Neill. Order-preserving symmetric encryption. In *EUROCRYPT*, 2009.

[BCO11] Alexandra Boldyreva, Nathan Chenette, and Adam O'Neill. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In *CRYPTO*, 2011.

[BCP13] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. *IACR Cryptology ePrint Archive*, 2013:650, 2013.

[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 *CRYPTO*, 2001.

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

[BO13] Mihir Bellare and Adam O'Neill. Semantically-secure functional encryption: Possibility results, impossibility results and the quest for a general definition. In *CANS*, 2013.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *TCC*, 2011.

[BW06] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *CRYPTO*, pages 290–307, 2006.

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, 2007.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *ASIACRYPT*, 2013.

[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 *CRYPTO (2)*, 2013.

[CKKC13] Seung Geol Choi, Jonathan Katz, Ranjit Kumaresan, and Carlos Cid. Multi-client non-interactive verifiable computation. In *TCC*, 2013.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. In *STOC*, pages 542–552, 1991.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.

[DNR+09] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil P. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *STOC*, pages 381–390, 2009.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

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

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. In *CRYPTO*, 2013.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *STOC*, pages 467–476, 2013.

[GJKS13] Vipul Goyal, Abhishek Jain, Venkata Koppula, and Amit Sahai. How to compute randomized functions on encrypted data. *IACR Cryptology ePrint Archive*, 2013, 2013.

[GKP+13a] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run turing machines on encrypted data. In *CRYPTO (2)*, pages 536–553, 2013.

[GKP+13b] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *STOC*, 2013.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM Conference on Computer and Communications Security*, 2006.

[GS02] Craig Gentry and Alice Silverberg. Hierarchical id-based cryptography. In *ASIACRYPT*, pages 548–566, 2002.

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

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In *STOC*, 2013.

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

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, 2008.

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

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC*, pages 427–437, 1990.

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

[PR12] Omkant Pandey and Yannis Rouselakis. Property preserving symmetric encryption. In *EUROCRYPT*, 2012.

[PRV12] Bryan Parno, Mariana Raykova, and Vinod Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In *TCC*, 2012.

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *FOCS*, pages 543–553, 1999.

[SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In *ACM Conference on Computer and Communications Security*, pages 463–472, 2010.

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

[SW13] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. *IACR Cryptology ePrint Archive*, 2013:454, 2013.

A Completing sel-IND Security Proof for $\mathcal{FE}_{\mathsf{I}}$

Lemma 28 ($\mathsf{H}_0 \stackrel{c}{\equiv} \mathsf{H}_1$)**.** *Assuming that* Com *is a (computationally) hiding commitment scheme, the outputs of experiments* H_0 *and* H_1 *are computationally indistinguishable.*

Proof. Recall that the only difference between H_0 and H_1 is the manner in which the commitments $\{Z_1^{i,j}\}$ are computed: in H_0, every $Z_1^{i,j}$ is a commitment to the all zeros string 0^{len}, while in H_1, $Z_1^{i,j}$ is a commitment to $\hat{c}_1^{i,j} \| \hat{c}_2^{i,j}$. Further, note that the randomness used to compute $Z_1^{i,j}$ is not used elsewhere in the experiment. Then, by a standard hybrid argument, the indistinguishability of H_0 and H_1 follows from the computational hiding property of Com. We omit the details. □

Lemma 29 ($\mathsf{H}_1 \stackrel{c}{\equiv} \mathsf{H}_2$)**.** *Assuming that* $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ *is witness indistinguishable, the outputs of experiments* H_1 *and* H_2 *are computationally indistinguishable.*

Proof. Recall that the only difference between H_1 and H_2 is the manner in which the proof strings $\hat{\pi}^{i,j}$ in challenge ciphertexts $\hat{\mathsf{CT}}_{i,j}$ are computed: in H_1, every $\hat{\pi}^{i,j}$ is computed using the *real* witness, while in H_2, $\hat{\pi}^{i,j}$ is computed using the *trapdoor* witness. Then, by a standard hybrid argument, the indistinguishability of H_1 and H_2 follows from the witness indistinguishability property of the NIWI proof system. □

Lemma 30 ($\mathsf{H}_2 \stackrel{c}{\equiv} \mathsf{H}_3$)**.** *Assuming that* Com *is a (computationally) hiding commitment scheme, the outputs of experiments* H_2 *and* H_3 *are computationally indistinguishable.*

Proof. Recall that the only difference between H_2 and H_3 is the manner in which the commitments $\{Z_2^i\}$ are computed: in H_2, every Z_2^i, where $i \in \mathrm{I}$, is a commitment to 0, while in H_3, Z_2^i is a commitment to 1. Further, note that the randomness used to compute Z_2^i is not used anywhere else in the experiment. Then, by a standard hybrid argument, the indistinguishability of H_2 and H_3 follows from the computational hiding property of Com. □

Lemma 31 ($\mathsf{H}_3 \stackrel{c}{\equiv} \mathsf{H}_4$)**.** *Assuming that* $\mathsf{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ *is a semantically-secure public-key encryption scheme, the outputs of experiments* H_3 *and* H_4 *are computationally indistinguishable.*

Proof. Recall that the only difference between H_3 and H_4 is the manner in which the second ciphertexts $\hat{c}_2^{i,j}$ in the challenge ciphertexts $\hat{\mathsf{CT}}_{i,j}$ are computed: in H_3, $\hat{c}_2^{i,j}$ is an encryption of the challenge message $x_{i,j}$, while in H_4, $\hat{c}_2^{i,j}$ is an encryption of 0. Further, note that neither the randomness $s_2^{i,j}$ used to compute $\hat{c}_2^{i,j}$, nor the secret key sk_2 is used anywhere else in the experiment. Then, by a standard hybrid argument, the indistinguishability of H_3 and H_4 follows from the semantic security of PKE. □

Lemma 32 ($\mathsf{H}_4 \stackrel{c}{\equiv} \mathsf{H}_5$)**.** *Assuming that* $i\mathcal{O}$ *is an indistinguishability obfuscator,* Com *is perfectly binding and* $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ *is a proof system, the outputs of the experiments* H_4 *and* H_5 *are computationally indistinguishable.*

Proof. We prove the lemma for the simplified case where the adversary makes a single key query f. We remark that by a standard hybrid argument, the proof can be easily extended to the more general case where the adversary makes $\mathsf{poly}(k)$ number of key queries.

Now, note that the only difference between H_4 and H_5 is the manner in which the secret key SK_f for the key query f is computed: in experiment H_4, SK_f is an indistinguishability obfuscation of G_f, while in H_5, SK_f is an indistinguishability obfuscation of $\mathsf{Sim.G}'_f$. Now, if G_f and G'_f have the same output behavior on *all* input points, then the computational indistinguishability of H_4 and H_5 follows immediately from the indistinguishability of $i\mathcal{O}(\mathsf{G}_f)$ and $i\mathcal{O}(\mathsf{G}'_f)$. Thus, all that remains to prove is that for all inputs z, $\mathsf{G}_f(z) = \mathsf{G}'_f(z)$.

Towards that end, we first assume without loss of generality that the encryption scheme $\mathsf{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ does not have any decryption error. We make the following claim:

Claim 33. *For any input* z, $\mathsf{G}_f(z) = \bot$ *iff* $\mathsf{G}'_f(z) = \bot$.

Proof. Let $z = (\mathsf{CT}_1, \ldots, \mathsf{CT}_n)$ be any input to G_f and G'_f. For every $i \in [n]$, let $\mathsf{CT}_i = (c_{i,1}, c_{i,2}, \pi_i)$. Note that both $\mathsf{Sim.G}_f$ and $\mathsf{Sim.G}'_f$ output $\bot$ on input z iff there exists $i \in [n]$ such that $\mathsf{Verify}(\mathsf{crs}, y_i, \pi_i) = 0$, where $y_i = (c_{i,1}, c_{i,2}, \mathsf{pk}_1, \mathsf{pk}_2, \{Z_1^{i,j}\}, Z_2^i)$ is the statement corresponding to the NIWI proof π_i. The claim follows. □

Following the above claim, we shall call an input z to G_f and G'_f to be a *valid* input if $\mathsf{G}_f(z) \neq \bot$ (and $\mathsf{G}'_f(z) \neq \bot$). We now demonstrate that the outputs of G_f and G'_f differ on a valid input z only if z satisfies some specific properties. Later, we will rely on the binding property of Com and the statistical soundness of the NIWI proof system to show that such an input z does not exist, thus completing the proof.

Claim 34. *Let* $\{\hat{\mathsf{CT}}_{1,j}, \ldots, \hat{\mathsf{CT}}_{n,j}\}_{j=1}^q$ *denote the challenge ciphertexts given to the adversary, where every* $\hat{\mathsf{CT}}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$. *Then, for every* valid *input* $z = (\mathsf{CT}_1, \ldots, \mathsf{CT}_n)$ *to* G_f *and* G'_f *such that* $\mathsf{G}_f(z) \neq \mathsf{G}'_f(z)$*, there exists* $i \in [n]$, $\mathsf{CT}_i = (c_{i,1}, c_{i,2}, \pi_i)$ *in* z *such that one of the following two cases holds:*

Case 1: *If* $i \in \mathrm{I}$*, then* $c_{i,1}$ *and* $c_{i,2}$ *are encryptions of different messages, and for every* $j \in [q]$*, either* $\hat{c}_{i,1} \neq \hat{c}_1^{i,j}$ *or* $c_{i,2} \neq \hat{c}_2^{i,j}$.

Case 2: *If* $i \in \mathsf{N} \setminus \mathrm{I}$*, then for every* $j \in [q]$*, either* $c_{i,1} \neq \hat{c}_1^{i,j}$ *or* $c_{i,2} \neq \hat{c}_2^{i,j}$.

Proof. Suppose that the claim is false. That is, there exists a valid input $z^* = (\mathsf{CT}_1^*, \ldots, \mathsf{CT}_n^*)$ such that $\mathsf{G}_f(z^*) \neq \mathsf{G}'_f(z^*)$, yet z satisfies the following conditions:

Condition A: For every $i \in \mathrm{I}$, $\mathsf{CT}_i^* = (c_{i,1}^*, c_{i,2}^*, \pi_i^*)$ is such that:

1. Either there exists $j_i \in [q]$ such that $c^*_{i,1} = \hat{c}_1^{i,j_i}$ and $c^*_{i,2} = \hat{c}_2^{i,j_i}$, or
2. $c^*_{i,1}$ and $c^*_{i,2}$ are encryptions of the *same* message. Let x'_i denote this message.

Condition B: For every $i \in \mathsf{N}\setminus\mathrm{I}$, $\mathsf{CT}^*_i = (c^*_{i,1}, c^*_{i,2}, \pi^*_i)$, there exists $j_i \in [q]$ such that $c^*_{i,1} = \hat{c}_1^{i,j_i}$ and $c^*_{i,2} = \hat{c}_2^{i,j_i}$.

Let us inspect the outputs of G_f and G'_f on the input z^*. We have that:

$$\mathsf{G}_f(z^*) = f\left(\left\langle \left\{\mathsf{PKE.Dec}\left(\mathsf{sk}_1, \hat{c}_1^{i,j_i}\right)\right\}_{i\in\mathsf{N}\setminus\mathrm{I}}, \left\{\mathsf{PKE.Dec}\left(\mathsf{sk}_1, c^*_{i,1}\right)\right\}_{i\in\mathrm{I}}\right\rangle\right),$$

$$\mathsf{G}'_f(z^*) = f\left(\left\langle \left\{\mathsf{PKE.Dec}\left(\mathsf{sk}_2, \hat{c}_2^{i,j_i}\right)\right\}_{i\in\mathsf{N}\setminus\mathrm{I}}, \left\{\mathsf{PKE.Dec}\left(\mathsf{sk}_2, c^*_{i,2}\right)\right\}_{i\in\mathrm{I}}\right\rangle\right),$$

where for $\ell \in [2]$, $\left\langle \left\{\mathsf{PKE.Dec}\left(\mathsf{sk}_\ell, \hat{c}_\ell^{i,j_i}\right)\right\}_{i\in\mathsf{N}\setminus\mathrm{I}}, \left\{\mathsf{PKE.Dec}\left(\mathsf{sk}_\ell, c^*_{i,\ell}\right)\right\}_{i\in\mathrm{I}}\right\rangle$ denotes the "arrangement" of the values $\left\{\mathsf{PKE.Dec}\left(\mathsf{sk}_\ell, \hat{c}_\ell^{i,j_i}\right)\right\}_{i\in\mathsf{N}\setminus\mathrm{I}}$, $\left\{\mathsf{PKE.Dec}\left(\mathsf{sk}_\ell, c^*_{i,\ell}\right)\right\}_{i\in\mathrm{I}}$ according to their input positions in f. Now, let $\mathrm{I}' \subseteq \mathrm{I}$ be such that for every $i \in \mathrm{I}'$, $\mathsf{CT}^*_i = (c^*_{i,1}, c^*_{i,2}, \pi^*_i)$ satisfies the condition $A(2)$. Thus, for every $i \in \mathrm{I}\setminus\mathrm{I}'$, $\mathsf{CT}^*_i = (c^*_{i,1}, c^*_{i,2}, \pi^*_i)$ satisfies the condition $A(1)$. Then, we have:

$$\mathsf{G}_f(z^*) = f\left(\left\langle \{x^0_{i,j_i}\}_{i\in\mathsf{N}\setminus\mathrm{I}'}, \{x'_i\}_{i\in\mathrm{I}'}\right\rangle\right) \;;\; \mathsf{G}'_f(z^*) = f\left(\left\langle \{x^1_{i,j_i}\}_{i\in\mathsf{N}\setminus\mathrm{I}'}, \{x'_i\}_{i\in\mathrm{I}'}\right\rangle\right),$$

where $\vec{X}^0 = \{x^0_{1,j}, \ldots, x^0_{n,j}\}_{j=1}^q$ and $\vec{X}^1 = \{x^1_{1,j}, \ldots, x^1_{n,j}\}_{j=1}^q$ are the challenge messages.

Now, it follows from the I-Compatibility property in the IND-security definition (see Definition 3) that $\mathsf{G}_f(z^*) = \mathsf{G}'_f(z^*)$, which is a contradiction. □

Completing the proof of Lemma 32. We now prove that for every valid input z, $\mathsf{G}_f(z) \neq \mathsf{G}'_f(z)$. For the sake of contradiction, suppose not. That is, let z^* be a valid input such that $\mathsf{G}_f(z^*) = \mathsf{G}'_f(z^*)$. Following Claim 34, fix $i \in [n]$, $\mathsf{CT}^*_i = (c^*_{i,1}, c^*_{i,2}, \pi^*_i)$ in z^* to be such that either Case 1 or Case 2 holds.

First observe that since z^* is a valid input, we have that $\mathsf{Verify}(\mathsf{crs}, y^*_i, \pi^*_i) = 1$, where $y^*_i = (c^*_{i,1}, c^*_{i,2}, \mathsf{pk}_1, \mathsf{pk}_2, \{Z_1^{i,j}\}, Z_1^i)$ is the statement corresponding to the proof string π^*_i. Then, since $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ is a statistically sound proof system, it follows that the statement y^*_i must be *true*, i.e., either there exists a *real* witness or a *trapdoor* witness for y^*_i (see Section 4 for the definitions of real and trapdoor witnesses). We now consider the two cases:

Case 1. $i \in \mathrm{I}$**:** Since $c^*_{i,1}$ and $c^*_{i,2}$ are encryptions of different messages, there does not exist a *real* witness for y^*_i. Then, suppose that there exists a *trapdoor* witness $w_{\mathsf{trap}} = (j, r_1^{i,j})$ for y^*_i. That is, suppose that $\exists j \in [q]$ and randomness $r_1^{i,j}$ such that $Z_1^{i,j} = \mathsf{Com}(c^*_{i,1}, c^*_{i,2}; r_1^{i,j})$. However, note that in experiments H_4 and H_5, $Z_1^{i,j}$ is computed as a commitment to $\hat{c}_1^{i,j} \| \hat{c}_2^{i,j}$. Since Com is perfectly binding and either $c^*_{i,1} \neq \hat{c}_1^{i,j}$ or $c^*_{i,2} \neq \hat{c}_2^{i,j}$, we obtain a contradiction.

Case 2. $i \in \mathsf{N}\setminus\mathrm{I}$**:** First observe that since Z_2^i is computed as a commitment to 1 in experiments H_4 and H_5, it follows from the perfect binding property of Com that there does not exist a *real* witness for y^*_i. Then, suppose that there exists a *trapdoor* witness $w_{\mathsf{trap}} = (j, r_1^{i,j})$ for y^*_i. That is, suppose that $\exists j \in [q]$ and randomness $r_1^{i,j}$ such that $Z_1^{i,j} = \mathsf{Com}(c^*_{i,1}, c^*_{i,2}; r_1^{i,j})$. However, note that in experiments H_4 and H_5, $Z_1^{i,j}$ is computed as a commitment to $\hat{c}_1^{i,j} \| \hat{c}_2^{i,j}$. Since Com is perfectly binding and either $c^*_{i,1} \neq \hat{c}_1^{i,j}$ or $c^*_{i,2} \neq \hat{c}_2^{i,j}$, we obtain a contradiction. □

Lemma 35 ($\mathsf{H}_5 \stackrel{c}{\equiv} \mathsf{H}_6$)**.** *Assuming that* $\mathsf{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ *is a semantically-secure public-key encryption scheme, the outputs of experiments* H_5 *and* H_6 *are computationally indistinguishable.*

Proof. The proof follows in the same manner as Lemma 31. □

Lemma 36 ($\mathsf{H}_6 \stackrel{c}{\equiv} \mathsf{H}_7$)**.** *Assuming that $i\mathcal{O}$ is an indistinguishability obfuscator,* $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ *is a proof system, and* Com *is perfectly binding, the outputs of experiments* H_6 *and* H_7 *are computationally indistinguishable.*

Proof. The proof follows in the same manner as Lemma 32. □

Lemma 37 ($\mathsf{H}_7 \stackrel{c}{\equiv} \mathsf{H}_8$)**.** *Assuming that* Com *is a (computationally) hiding commitment scheme, the outputs of experiments* H_7 *and* H_8 *are computationally indistinguishable.*

Proof. The proof follows in the same manner as Lemma 30. □

Lemma 38 ($\mathsf{H}_8 \stackrel{c}{\equiv} \mathsf{H}_9$)**.** *Assuming that* $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ *is witness indistinguishable, the outputs of experiments* H_8 *and* H_9 *are computationally indistinguishable.*

Proof. The proof follows in the same manner as Lemma 29. □

Lemma 39 ($\mathsf{H}_9 \stackrel{c}{\equiv} \mathsf{H}_{10}$)**.** *Assuming that* Com *is a (computationally) hiding commitment scheme, the outputs of experiments* H_9 *and* H_{10} *are computationally indistinguishable.*

Proof. The proof follows in the same manner as Lemma 28. □

B Completing SIM Security Proof for $\mathcal{FE}_{\mathsf{I}}$

We now describe a series of hybrid experiments $\mathsf{H}_0, \ldots, \mathsf{H}_8$, where H_0 corresponds to the real world and H_8 corresponds to the ideal world experiment. For every i, we will prove that the output of H_i is computationally indistinguishable from the output of H_{i+1}.

Hybrid H_0: This is the real experiment.

Hybrid H_1: This experiment is the same as H_0 except in the manner in which the key queries of the adversary are answered. Let $\{x_{1,j}, \ldots, x_{n,j}\}_{j=1}^{q} \leftarrow M$ be the challenge messages. Then, whenever the adversary makes a key query f, we perform the following steps:

- Query the trusted party TP on function f. For every $j_1, \ldots, j_n \in [q]$, the trusted party computes and returns the function output $\mathsf{out}[j_1, \ldots, j_n] = f(x_{1,j_1}, \ldots, x_{n,j_n})$.
- Compute the secret key SK_f for function f as $\mathsf{SK}_f \leftarrow i\mathcal{O}(\mathsf{Sim.G}_f)$, where $\mathsf{Sim.G}_f$ is as described in Figure 2.

For every $i \in [n]$, $j \in [q]$, let $\hat{\mathsf{CT}}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$ denote the challenge ciphertext computed by the experiment. Then, note that $\mathsf{Sim.G}_f$ has the master secret key MSK, the ciphertext pairs $\{\hat{c}_1^{i,j}, \hat{c}_2^{i,j}\}$ and the outputs $\{\mathsf{out}[j_1, \ldots, j_n]\}$ hardwired in it.

Hybrid H_2: This experiment is the same as H_1 except that the setup algorithm computes the commitments $\{Z_1^{i,j}\}$ in the following manner: let the challenge ciphertext $\hat{\mathsf{CT}}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$. Then, $Z_1^{i,j} \leftarrow \mathsf{Com}(\hat{c}_1^{i,j} \| \hat{c}_2^{i,j})$.

Hybrid H_3: This experiment is the same as H_2 except that in every challenge ciphertext $\hat{\mathsf{CT}}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$, the proof string $\hat{\pi}^{i,j}$ is computed using the *trapdoor* witness.

Hybrid H_4: This experiment is the same as H_3 except that the setup algorithm computes every Z_2^i as a commitment to 1 (instead of 0). That is, for every $i \in [n]$, $Z_2^i \leftarrow \mathsf{Com}(1)$.

Hybrid H_5: This experiment is the same as H_4 except that in every challenge ciphertext $\hat{\mathsf{CT}}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$, the *second* ciphertext $\hat{c}_2^{i,j}$ is an encryption of zeros, i.e., $\hat{c}_2^{i,j} \leftarrow \mathsf{FE.Enc}(\mathsf{EK}_i, 0^k)$.

Hybrid H_6: This experiment is the same as H_5 except that for every key query f, the corresponding secret key SK_f is computed as $\mathsf{SK}_f \leftarrow i\mathcal{O}(\mathsf{Sim.G}'_f)$ where $\mathsf{Sim.G}'_f$ is the same as the function $\mathsf{Sim.G}_f$ except that:

1. It has secret key sk_2 hardwired instead of sk_1.
2. It decrypts the *second* component of each input ciphertext using sk_2. More concretely, in step $1(c)$, plaintext x'_i is computed as $x'_i \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}_2, c_{i,2})$.

Hybrid H_7: This experiment is the same as H_6 except that in every challenge ciphertext $\hat{\mathsf{CT}}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$, the *first* ciphertext $\hat{c}_1^{i,j}$ is an encryption of zeros, i.e., $\hat{c}_1^{i,j} \leftarrow \mathsf{FE.Enc}(\mathsf{EK}_i, 0^k)$.

Hybrid H_8: This experiment is the same as H_7 except that for every key query f, the corresponding secret key SK_f is computed as $\mathsf{SK}_f \leftarrow i\mathcal{O}(\mathsf{Sim.G}_f)$. Note that this is the ideal world experiment.

This completes the description of the hybrid experiments. We note that the proof of indistinguishability of the hybrid experiments described above bear much similarity to the proof of IND security (Section 4.1). Therefore, to avoid repetition, below we only focus on the key hybrids that differ from the IND security case. Specifically, below, we prove indistinguishability of hybrid experiments H_0 and H_1, and then H_5 and H_6. For details on the rest of the proof, see Appendix A.

Lemma 40 ($\mathsf{H}_0 \stackrel{c}{\equiv} \mathsf{H}_1$)**.** *Assuming that $i\mathcal{O}$ is an indistinguishability obfuscator, the outputs of experiments H_0 and H_1 are computationally indistinguishable.*

Proof. We prove the lemma for the simplified case where the adversary makes a single key query f. By a standard hybrid argument, the proof can be easily extended to the more general case where the adversary makes $\mathsf{poly}(k)$ number of key queries.

Now, note that the only difference between H_0 and H_1 is the manner in which the secret key SK_f for the key query f is computed: in experiment H_0, SK_f is an indistinguishability obfuscation of G_f, while in H_1, SK_f is an indistinguishability obfuscation of $\mathsf{Sim.G}_f$. Now, if G_f and $\mathsf{Sim.G}_f$ have the same output behavior on *all* input points, then the computational indistinguishability of H_0 and H_1 follows immediately from the indistinguishability of $i\mathcal{O}(\mathsf{G}_f)$ and $i\mathcal{O}(\mathsf{Sim.G}_f)$. Thus, all that remains to prove is that for all inputs z, $\mathsf{G}_f(z) = \mathsf{Sim.G}_f(z)$.

Towards that end, let $\{\hat{\mathsf{CT}}_{1,j}, \ldots, \hat{\mathsf{CT}}_{n,j}\}_{j=1}^q$ denote the challenge ciphertexts computed in experiments H_1 and H_2, where every $\hat{\mathsf{CT}}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$. We say that an input $z = (\mathsf{CT}_1, \ldots, \mathsf{CT}_n)$ to G_f and $\mathsf{Sim.G}_f$ is *special* if for every $\mathsf{CT}_i = (c_{i,1}, c_{i,2}, \pi_i)$:

- The proof π_i is accepting, and
- There exists $j_i \in [q]$ s.t. $c_{i,1} = \hat{c}_1^{i,j_i}$ and $c_{i,2} = \hat{c}_2^{i,j_i}$.

Further, we call $(j_1, \ldots, j_n)$ to be the "index set" of z.

Now note that the only difference between the functions G_f and $\mathsf{Sim.G}_f$ is that on a special input z with index set $(j_1, \ldots, j_n)$, $\mathsf{Sim.G}_f$ skips the usual decryption step and directly outputs the value $\mathsf{out}[j_1, \ldots, j_n]$ hardwired in its description. Recall that (by definition) $\mathsf{out}[j_1, \ldots, j_n] = f(x_{1,j_1}, \ldots, x_{n,j_n})$ where $\{x_{1,j}, \ldots, x_{n,j}\}_{j=1}^q$ denote the challenge messages. However, on such an input z, by performing the decryption step, G_f obtains the messages $(x_{1,j_1}, \ldots, x_{n,j_n})$ and therefore its output is $f(x_{1,j_1}, \ldots, x_{n,j_n})$ as well. □

Lemma 41 ($\mathsf{H}_5 \stackrel{c}{\equiv} \mathsf{H}_6$)**.** *Assuming that $i\mathcal{O}$ is an indistinguishability obfuscator,* Com *is perfectly binding, and* $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ *is a proof system, the outputs of experiments* H_5 *and* H_6 *are computationally indistinguishable.*

Proof. We prove the lemma for the simplified case where the adversary makes a single key query f. By a standard hybrid argument, our proof can be easily extended to the more general case where the adversary makes $\mathsf{poly}(k)$ number of key queries.

Now, note that the only difference between H_5 and H_6 is the manner in which the secret key SK_f for the key query f is computed: in experiment H_5, SK_f is an indistinguishability obfuscation of $\mathsf{Sim}.\mathsf{G}_f$, while in H_6, SK_f is an indistinguishability obfuscation of $\mathsf{Sim}.\mathsf{G}'_f$. Now, if $\mathsf{Sim}.\mathsf{G}_f$ and $\mathsf{Sim}.\mathsf{G}'_f$ have the same output behavior on *all* input points, then the computational indistinguishability of H_5 and H_6 follows immediately from the indistinguishability of $i\mathcal{O}(\mathsf{Sim}.\mathsf{G}_f)$ and $i\mathcal{O}(\mathsf{Sim}.\mathsf{G}'_f)$. Thus, all that remains to prove is that for all inputs z, $\mathsf{Sim}.\mathsf{G}_f(z) = \mathsf{Sim}.\mathsf{G}'_f(z)$.

Towards that end, we first assume without loss of generality that the encryption scheme $\mathsf{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ does not have any decryption error. We make the following claim:

Claim 42. *For any input z, $\mathsf{Sim}.\mathsf{G}_f(z) = \bot$ iff $\mathsf{Sim}.\mathsf{G}'_f(z) = \bot$.*

Proof. Let $z = (\mathsf{CT}_1, \ldots, \mathsf{CT}_n)$ be any input to $\mathsf{Sim}.\mathsf{G}_f$ and $\mathsf{Sim}.\mathsf{G}'_f$. For every $i \in [n]$, let $\mathsf{CT}_i = (c_{i,1}, c_{i,2}, \pi_i)$. Note that both $\mathsf{Sim}.\mathsf{G}_f$ and $\mathsf{Sim}.\mathsf{G}'_f$ output $\bot$ on input z iff there exists $i \in [n]$ such that $\mathsf{Verify}(\mathsf{crs}, y_i, \pi_i) = 0$, where $y_i = (c_{i,1}, c_{i,2}, \mathsf{pk}_1, \mathsf{pk}_2, \{Z_1^{i,j}\}, Z_2^i)$ is the statement corresponding to the NIWI proof π_i. The claim immediately follows. □

Following the above claim, we shall call an input z to $\mathsf{Sim}.\mathsf{G}_f$ and $\mathsf{Sim}.\mathsf{G}'_f$ to be a *valid* input if $\mathsf{Sim}.\mathsf{G}_f(z) \neq \bot$ (and $\mathsf{Sim}.\mathsf{G}'_f(z) \neq \bot$). We make the following claim regarding valid inputs:

Claim 43. *Let $\{\hat{\mathsf{CT}}_{1,j}, \ldots, \hat{\mathsf{CT}}_{n,j}\}_{j=1}^q$ denote the challenge ciphertexts given to the adversary, where every $\hat{\mathsf{CT}}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$. Let $z = (\mathsf{CT}_1, \ldots, \mathsf{CT}_n)$ denote a* valid *input to $\mathsf{Sim}.\mathsf{G}_f$ and $\mathsf{Sim}.\mathsf{G}'_f$. Then, for every $\mathsf{CT}_i = (c_{i,1}, c_{i,2}, \pi_i)$, there exists $j_i \in [q]$ s.t. $c_{i,1} = \hat{c}_1^{i,j_i}$ and $c_{i,2} = \hat{c}_2^{i,j_i}$.*

Proof. Suppose that the claim is false. That is, for a valid input $z = (\mathsf{CT}_1, \ldots, \mathsf{CT}_n)$, $\exists\ \mathsf{CT}_i = (c_{i,1}, c_{i,2}, \pi_i)$ s.t. $\forall\ j \in [q]$, either $c_{i,1} \neq \hat{c}_1^{i,j}$ or $c_{i,2} \neq \hat{c}_2^{i,j}$. Now, since z is a valid input, we have that $\mathsf{Verify}(\mathsf{crs}, y_i, \pi_i) = 1$, where $y_i = (c_{i,1}, c_{i,2}, \mathsf{pk}_1, \mathsf{pk}_2, \{Z_1^{i,j}\}, Z_2^i)$ is the statement corresponding to the NIWI proof π_i. Then, since $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ is a statistically sound proof system, it follows that the statement y_i must be *true*. We consider the following two cases:

Case 1: The ciphertexts $c_{i,1}$ and $c_{i,2}$ are encryptions of the same message and there exists randomness r_2^i s.t. $Z_2 \leftarrow \mathsf{Com}(0; r_2^i)$. However, note that in experiments H_5 and H_6, Z_2^i is computed as a commitment to 1. Since Com is a perfectly binding commitment scheme, we obtain a contradiction.

Case 2: $\exists j \in [q]$ and randomness $r_1^{i,j}$ such that $Z_1^{i,j} = \mathsf{Com}(c_{i,1} \| c_{i,2}; r_1^{i,j})$. However, note that in experiments H_5 and H_6, $Z_1^{i,j}$ is computed as a commitment to $\hat{c}_1^{i,j} \| \hat{c}_2^{i,j}$. Since Com is a perfectly binding commitment scheme and either $c_{i,1} \neq \hat{c}_1^{i,j}$ or $c_{i,2} \neq \hat{c}_2^{i,j}$, we obtain a contradiction.

This completes the proof of the above claim. □

Completing the proof of Lemma 41. Following Claim 42, we only need to prove that for every valid input z, $\mathsf{Sim}.\mathsf{G}_f(z) = \mathsf{Sim}.\mathsf{G}'_f(z)$. Now, let $z = (\mathsf{CT}_1, \ldots, \mathsf{CT}_n)$ be any valid input to $\mathsf{Sim}.\mathsf{G}_f$ and $\mathsf{Sim}.\mathsf{G}'_f$. From Claim 49, we have that for every $i \in [n]$, there exists $j_i \in [q]$ such that $\mathsf{CT}_i = \hat{c}_1^{i,j_i}, \hat{c}_2^{i,j_i}, \pi_i$. Then, note that on such an input $z = (\mathsf{CT}_1, \ldots, \mathsf{CT}_n)$, both $\mathsf{Sim}.\mathsf{G}_f$ and $\mathsf{Sim}.\mathsf{G}'_f$ output the *same* (programmed) value, i.e., $\mathsf{out}[j_1, \ldots, j_n]$.

This completes the proof of Lemma 41. □

C Proving IND Security for $\mathcal{FE}_{\mathsf{II}}$

We now prove that the proposed scheme $\mathcal{FE}_{\mathsf{II}}$ is $(t, \mathsf{poly}(k))$-IND-secure for any $t \leq n$ and arbitrary $\mathsf{poly}(k)$ number of message queries. We will prove security via a series of hybrid experiments $\mathsf{H}_0, \ldots, \mathsf{H}_8$, where H_0 (resp., H_8) corresponds to the real world experiment with challenge bit $b = 0$ (resp., $b = 1$).

Hybrid H_0: This is the real experiment with challenge bit $b = 0$.

Hybrid H_1: This experiment is the same as H_0 except that the setup algorithm computes a "simulated" CRS for the simulation-sound NIZK proof system, i.e., the CRS is computed as $(\mathsf{crs}, \tau) \leftarrow \mathsf{Sim.CRSGen}(1^k)$.

Hybrid H_2: This experiment is the same as H_1 except that in every challenge ciphertext $\hat{\mathsf{CT}}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$, $\hat{\pi}^{i,j}$ is computed as a simulated proof, i.e., $\hat{\pi}^{i,j} \leftarrow \mathsf{Sim.Prove}(\mathsf{crs}, \tau, y_{i,j})$ where the statement $y_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \mathsf{pk}_1, \mathsf{pk}_2, Z_i)$.

Hybrid H_3: This experiment is the same as H_2 except that for every $i \in \mathsf{N} \setminus \mathrm{I}$, the setup algorithm computes every Z_i as a commitment to 1 (instead of 0), i.e., $Z_i \leftarrow \mathsf{Com}(1)$.

Hybrid H_4: This experiment is the same as H_3 except that in every challenge ciphertext $\hat{\mathsf{CT}}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$, the *second* ciphertext $\hat{c}_2^{i,j}$ is an encryption of the challenge message $x_{i,j}^1$ (as opposed to $x_{i,j}^0$), i.e., $\hat{c}_2^{i,j} \leftarrow \mathsf{FE.Enc}(\mathsf{EK}_i, x_{i,j}^1)$.

Hybrid H_5: This experiment is the same as H_4 except that for every key query f, the corresponding secret key SK_f is computed as $\mathsf{SK}_f \leftarrow \mathsf{di}\mathcal{O}(\mathcal{H}'_f)$ where $\mathcal{H}'_f$ is the same as the function $\mathcal{H}_f$ except that:

1. It has secret key sk_2 hardwired instead of sk_1.
2. It decrypts the *second* component of each input ciphertext using sk_2. More concretely, in step $1(c)$, plaintext x'_i is computed as $x'_i \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}_2, c_{i,2})$.

Hybrid H_6: This experiment is the same as H_5 except that in every challenge ciphertext $\hat{\mathsf{CT}}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$, the *first* ciphertext $\hat{c}_1^{i,j}$ is an encryption of the challenge message $x_{i,j}^1$ (as opposed to $x_{i,j}^0$), i.e., $\hat{c}_1^{i,j} \leftarrow \mathsf{FE.Enc}(\mathsf{EK}_i, x_{i,j}^1)$.

Hybrid H_7: This experiment is the same as H_6 except that for every key query f, the corresponding secret key SK_f is computed as $\mathsf{SK}_f \leftarrow \mathsf{di}\mathcal{O}(\mathcal{H}_f)$.

Hybrid H_8: This experiment is the same as H_7 except that the setup algorithm computes every Z_i as a commitment to 0.

Hybrid H_9: This experiment is the same as H_8 except that in every challenge ciphertext $\hat{\mathsf{CT}}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$, the proof string $\hat{\pi}^{i,j}$ is computed using the honest prover algorithm.

Hybrid H_{10}: This experiment is the same as H_9 except that the setup algorithm computes an "honest" CRS for the NIZK proof system, i.e., the CRS is computed as $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^k)$. Note that this is the real experiment with challenge bit $b = 1$.

This completes the description of the hybrids. We now prove their computational indistinguishability via a series of lemmas.

Lemma 44 ($\mathsf{H}_0 \stackrel{c}{\equiv} \mathsf{H}_1$)**.** *Assuming that* $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ *is a zero-knowledge argument system, the outputs of experiments* H_0 *and* H_1 *are computationally indistinguishable.*

Proof. This follows immediately from the fact that the distributions $\{\mathsf{CRSGen}(1^k)\}$ and $\{\mathsf{Sim.CRSGen}(1^k)\}$ are computationally indistinguishable. □

Lemma 45 ($\mathsf{H}_1 \stackrel{c}{\equiv} \mathsf{H}_2$)**.** *Assuming that* $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ *is a zero-knowledge argument system, the outputs of experiments* H_1 *and* H_2 *are computationally indistinguishable.*

Proof. Recall that the only difference between H_1 and H_2 is the manner in which the proof strings $\hat{\pi}^{i,j}$ in challenge ciphertexts $\hat{\mathsf{CT}}_{i,j}$ are computed: in H_1, every $\hat{\pi}^{i,j}$ is computed honestly using the witness, while in H_2, $\hat{\pi}^{i,j}$ is a simulated proof computed using the simulator for the NIZK argument system. Then, by a standard hybrid argument, the indistinguishability of H_2 and H_3 follows from the zero-knowledge property of the NIZK argument system. □

Lemma 46 ($\mathsf{H}_2 \stackrel{c}{\equiv} \mathsf{H}_3$)**.** *Assuming that* Com *is a computationally hiding commitment scheme, the outputs of experiments* H_2 *and* H_3 *are computationally indistinguishable.*

Proof. Recall that the only difference between H_2 and H_3 is the manner in which the commitment $\{Z_i\}_{i \in \mathsf{N} \setminus \mathrm{I}}$ are computed: in H_2, every Z_i is a commitment to 0, while in H_3, Z_i is a commitment to 1. Further, note that the randomness used to compute Z is not used anywhere else in the experiment. Then, the indistinguishability of H_2 and H_3 follows immediately from the computational hiding property of Com. □

Lemma 47 ($\mathsf{H}_3 \stackrel{c}{\equiv} \mathsf{H}_4$)**.** *Assuming that* $\mathsf{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ *is a semantically-secure public-key encryption scheme, the outputs of experiments* H_3 *and* H_4 *are computationally indistinguishable.*

Proof. Recall that the only difference between H_3 and H_4 is the manner in which the second ciphertexts $\hat{c}_2^{i,j}$ in the challenge ciphertexts $\hat{\mathsf{CT}}_{i,j}$ are computed: in H_3, $\hat{c}_2^{i,j}$ is an encryption of the challenge message $x_{i,j}^0$, while in H_4, $\hat{c}_2^{i,j}$ is an encryption of $x_{i,j}^1$. Further, note that nether the randomness $s_2^{i,j}$ used to compute $\hat{c}_2^{i,j}$ nor the secret key sk_2 is used anywhere else in the experiment. Then, by a standard hybrid argument, the indistinguishability of H_3 and H_4 follows from the semantic security of PKE. □

Lemma 48 ($\mathsf{H}_4 \stackrel{c}{\equiv} \mathsf{H}_5$)**.** *Assuming that* $\mathsf{di}\mathcal{O}$ *is a differing-inputs obfuscator,* Com *is perfectly binding and* $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ *is simulation-sound, the outputs of experiments* H_4 *and* H_5 *are computationally indistinguishable.*

Proof. We prove the lemma for the simplified case where the adversary makes a single key query f. This query could either be made by adversary $\mathcal{A}_1$ or $\mathcal{A}_2$. In the former case, we refer to f as a *non-adaptive* key query, while in the latter case, we refer to it as an *adaptive* key query. We remark that by a standard hybrid argument, the proof can be easily extended to the more general case where the adversary makes $\mathsf{poly}(k)$ number of (non-adaptive and adaptive) key queries.

Now, note that the only difference between H_4 and H_5 is the manner in which the secret key SK_f for the key query f is computed: in experiment H_4, SK_f is a differing-inputs obfuscation of

$\mathcal{H}_f$, while in H_5, SK_f is a differing-inputs obfuscation of $\mathcal{H}'_f$. It follows that if there exists a PPT adversary $\mathcal{A}$ that distinguishes between the outputs of H_4 and H_5 with non-negligible probability, then we can construct a PPT adversary $\mathcal{A}'$ that distinguishes between $\mathsf{di}\mathcal{O}(\mathcal{H}_f)$ and $\mathsf{di}\mathcal{O}(\mathcal{H}'_f)$ with non-negligible probability. Then, it follows from Definition 9 that for such an adversary $\mathcal{A}'$, there exists a PPT extractor algorithm E that on input $(\mathcal{H}_f, \mathcal{H}'_f)$ outputs an input value z^* such that $\mathcal{H}_f(z^*) \neq \mathcal{H}'_f(z^*)$. We will use E to contradict the simulation-soundness property of the NIZK argument system $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$.

Towards that end, let $z^* = (\mathsf{CT}^*_1, \ldots, \mathsf{CT}^*_n)$, where for every $i \in [n]$, $\mathsf{CT}^*_i = (c^*_{i,1}, c^*_{i,2}, \pi^*_i)$. Without loss of generality, we assume that every proof string π^*_i is *accepting*. This is because otherwise from the definition of $\mathcal{H}_f$ and $\mathcal{H}'_f$, we have that $\mathcal{H}_f(z^*) \neq \mathcal{H}'_f(z^*)$. We make the following claim about the input z^*.

Claim 49. *Let $\{\hat{\mathsf{CT}}_{1,j}, \ldots, \hat{\mathsf{CT}}_{n,j}\}_{j=1}^q$ denote the challenge ciphertexts in experiments H_3 and H_4, where every $\hat{\mathsf{CT}}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$. Then, there exists $i \in [n]$, $\mathsf{CT}^*_i = (c^*_{i,1}, c^*_{i,2}, \pi^*_i)$ in z^* such that one of the following two cases holds:*

Case 1: *If $i \in \mathrm{I}$, then $c^*_{i,1}$ and $c^*_{i,2}$ are encryptions of different messages, and for every $j \in [q]$, either $c^*_{i,1} \neq \hat{c}_1^{i,j}$ or $c^*_{i,2} \neq \hat{c}_2^{i,j}$.*

Case 2: *If $i \in \mathsf{N} \setminus \mathrm{I}$, then for every $j \in [q]$, either $c^*_{i,1} \neq \hat{c}_1^{i,j}$ or $c^*_{i,2} \neq \hat{c}_2^{i,j}$.*

Proof. Suppose that the claim is false. That is, the input $z^* = (\mathsf{CT}^*_1, \ldots, \mathsf{CT}^*_n)$ output by E is such that:

Condition A: For every $i \in \mathrm{I}$, $\mathsf{CT}^*_i = (c^*_{i,1}, c^*_{i,2}, \pi^*_i)$ is such that:

1. Either there exists $j_i \in [q]$ such that $c^*_{i,1} = \hat{c}_1^{i,j_i}$ and $c^*_{i,2} = \hat{c}_2^{i,j_i}$, or
2. $c^*_{i,1}$ and $c^*_{i,2}$ are encryptions of the *same* message. Let x'_i denote this message.

Condition B: For every $i \in \mathsf{N} \setminus \mathrm{I}$, $\mathsf{CT}^*_i = (c^*_{i,1}, c^*_{i,2}, \pi^*_i)$, there exists $j_i \in [q]$ such that $c^*_{i,1} = \hat{c}_1^{i,j_i}$ and $c^*_{i,2} = \hat{c}_2^{i,j_i}$.

Let us now inspect the outputs of $\mathcal{H}_f$ and $\mathcal{H}'_f$ on the input z^*. We have that:

$$\mathcal{H}_f(z^*) = f\left(\left\langle \left\{\mathsf{PKE.Dec}\left(\mathsf{sk}_1, \hat{c}_1^{i,j_i}\right)\right\}_{i\in\mathsf{N}\setminus\mathrm{I}}, \left\{\mathsf{PKE.Dec}\left(\mathsf{sk}_1, c^*_{i,1}\right)\right\}_{i\in\mathrm{I}} \right\rangle\right),$$

$$\mathcal{H}'_f(z^*) = f\left(\left\langle \left\{\mathsf{PKE.Dec}\left(\mathsf{sk}_2, \hat{c}_2^{i,j_i}\right)\right\}_{i\in\mathsf{N}\setminus\mathrm{I}}, \left\{\mathsf{PKE.Dec}\left(\mathsf{sk}_2, c^*_{i,2}\right)\right\}_{i\in\mathrm{I}} \right\rangle\right),$$

where for $\ell \in [2]$, $\left\langle \left\{\mathsf{PKE.Dec}\left(\mathsf{sk}_\ell, \hat{c}_\ell^{i,j_i}\right)\right\}_{i\in\mathsf{N}\setminus\mathrm{I}}, \left\{\mathsf{PKE.Dec}\left(\mathsf{sk}_\ell, c^*_{i,\ell}\right)\right\}_{i\in\mathrm{I}} \right\rangle$ denotes the "arrangement" of the values $\left\{\mathsf{PKE.Dec}\left(\mathsf{sk}_\ell, \hat{c}_\ell^{i,j_i}\right)\right\}_{i\in\mathsf{N}\setminus\mathrm{I}}$, $\left\{\mathsf{PKE.Dec}\left(\mathsf{sk}_\ell, c^*_{i,\ell}\right)\right\}_{i\in\mathrm{I}}$ according to their input positions in f. Now, let $\mathrm{I}' \subseteq \mathrm{I}$ be such that for every $i \in \mathrm{I}'$, $\mathsf{CT}^*_i = (c^*_{i,1}, c^*_{i,2}, \pi^*_i)$ satisfies the condition $A(2)$. Thus, for every $i \in \mathrm{I} \setminus \mathrm{I}'$, $\mathsf{CT}^*_i = (c^*_{i,1}, c^*_{i,2}, \pi^*_i)$ satisfies the condition $A(1)$. Then, we have:

$$\mathcal{H}_f(z^*) = f\left(\left\langle \{x^0_{i,j_i}\}_{i\in\mathsf{N}\setminus\mathrm{I}'}, \{x'_i\}_{i\in\mathrm{I}'} \right\rangle\right) \; ; \; \mathcal{H}'_f(z^*) = f\left(\left\langle \{x^1_{i,j_i}\}_{i\in\mathsf{N}\setminus\mathrm{I}'}, \{x'_i\}_{i\in\mathrm{I}'} \right\rangle\right),$$

where $\vec{X}^0 = \{x^0_{1,j}, \ldots, x^0_{n,j}\}_{j=1}^q$ and $\vec{X}^1 = \{x^1_{1,j}, \ldots, x^1_{n,j}\}_{j=1}^q$ are the challenge messages.

Now, regardless of whether f is a non-adaptive or adaptive key query, it follows from the I-Compatibility property in the IND-security definition (see Definition 3) that $\mathcal{H}_f(z^*) = \mathcal{H}'_f(z^*)$, which is a contradiction. □

Completing the proof of Lemma 48. Following the above claim, fix $\mathsf{CT}_i^* = (c_{i,1}^*, c_{i,2}^*, \pi_i^*)$ in z^* to be such that either Case 1 or Case 2 holds. Let $y_i^* = (c_{i,1}^*, c_{i,2}^*, \mathsf{pk}_1, \mathsf{pk}_2, Z_i)$ be the statement corresponding to the proof string π_i^*. Further, let $\hat{y}_{i,j}$ be the statement corresponding to the proof string $\hat{\pi}^{i,j}$ in challenge ciphertext $\mathsf{CT}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$.

We consider Case 1 and Case 2 separately.

Case 1. Since $c_{i,1}^*$ and $c_{i,2}^*$ are encryptions of different messages, we have that the statement y_i^* is *false*. Further, since for all $j \in [q]$, either $c_{i,1}^* \neq \hat{c}_1^{i,j}$ or $c_{i,2}^* \neq \hat{c}_2^{i,j}$, we have that $y_i^* \neq \hat{y}_{i,j}$. Then, we have that the output z^* of the extractor algorithm E includes an *accepting* proof for a new, false statement y_i^*. This contradicts the simulation-soundness property of the NIZK argument system $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$.

Case 2. Since Z_i is computed as a commitment to 1 in experiments H_3 and H_4, it follows from the perfect binding property of Com that the statement y_i^* is *false*. Further, since for all $j \in [q]$, either $c_{i,1}^* \neq \hat{c}_1^{i,j}$ or $c_{i,2}^* \neq \hat{c}_2^{i,j}$, we have that $y_i^* \neq \hat{y}_{i,j}$. Then, we have that the output z^* of the extractor algorithm E includes an *accepting* proof for a new, false statement y_i^*. This contradicts the simulation-soundness property of the NIZK argument system $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$. □

Lemma 50 ($\mathsf{H}_5 \stackrel{c}{\equiv} \mathsf{H}_6$)**.** *Assuming that* $\mathsf{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ *is a semantically-secure public-key encryption scheme, the outputs of experiments* H_5 *and* H_6 *are computationally indistinguishable.*

Proof. The proof follows in the same manner as Lemma 47. □

Lemma 51 ($\mathsf{H}_6 \stackrel{c}{\equiv} \mathsf{H}_7$)**.** *Assuming that* $\mathsf{di}\mathcal{O}$ *is a differing-inputs obfuscator,* Com *is perfectly binding and* $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ *is simulation-sound, the outputs of experiments* H_6 *and* H_7 *are computationally indistinguishable.*

Proof. The proof follows in the same manner as Lemma 48. □

Lemma 52 ($\mathsf{H}_7 \stackrel{c}{\equiv} \mathsf{H}_8$)**.** *Assuming that* Com *is a computationally hiding commitment scheme, the outputs of experiments* H_7 *and* H_8 *are computationally indistinguishable.*

Proof. The proof follows in the same manner as Lemma 46. □

Lemma 53 ($\mathsf{H}_8 \stackrel{c}{\equiv} \mathsf{H}_9$)**.** *Assuming that* $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ *is a* NIZK *argument system, the outputs of experiments* H_8 *and* H_9 *are computationally indistinguishable.*

Proof. The proof follows in the same manner as Lemma 45. □

Lemma 54 ($\mathsf{H}_9 \stackrel{c}{\equiv} \mathsf{H}_{10}$)**.** *Assuming that* $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ *is a* NIZK *argument system, the outputs of experiments* H_9 *and* H_{10} *are computationally indistinguishable.*

Proof. The proof follows in the same manner as Lemma 44. □

D NA-SIM-secure MI-FE

NA-SIM security for multi-input FE is defined similarly to definition 4, except that now the adversary is required to make key queries *before* (as opposed to after) choosing the challenge messages. More concretely, we define (t, p, q)-NA-SIM-secure functional encryption where (as earlier) t denotes the number of encryption keys known to the adversary and q denotes the number of challenge messages per encryption key. The new parameter p denotes the total number of non-adaptive key queries by the adversary. Below, we present the formal definition.

Definition 55 (NA-SIM Security)**.** *We say that a functional encryption scheme $\mathcal{FE}$ for n-ary functions $\mathcal{F}$ is* (t, p, q)-NA-SIM-secure *if for every PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$, there exists a PPT simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2)$ such that the outputs of the following two experiments are computationally indistinguishable:*

Experiment $\mathsf{REAL}_{\mathcal{A}}^{\mathcal{FE}}(1^k)$*:*
$(\mathrm{I}, \mathsf{st}_0) \leftarrow \mathcal{A}_0(1^k)$ *where* $|\mathrm{I}| = t$
$(\vec{\mathsf{EK}}, \mathsf{MSK}) \leftarrow \mathsf{FE.Setup}(1^k)$
$(\mathcal{M}, \mathsf{st}_1) \leftarrow \mathcal{A}_1^{\mathsf{FE.Keygen}(\mathsf{MSK}, \cdot)}(\mathsf{st}_0, \vec{\mathsf{EK}}_{\mathrm{I}})$
$\vec{X} \leftarrow \mathcal{M}$ *where* $\vec{X} = \{x_{1,j}, \ldots, x_{n,j}\}_{j=1}^{q}$
$\mathsf{CT}_{i,j} \leftarrow \mathsf{FE.Enc}(\mathsf{EK}_i, x_{i,j})\ \forall i \in [n],\ j \in [q]$
$\alpha \leftarrow \mathcal{A}_2(\vec{\mathsf{CT}}, \mathsf{st}_1)$
Output: $(\mathrm{I}, \mathcal{M}, \vec{X}, \{f_\ell\}, \alpha)$

Experiment $\mathsf{IDEAL}_{\mathcal{S}}^{\mathcal{FE}}(1^k)$*:*
$(\mathrm{I}, \mathsf{st}_0) \leftarrow \mathcal{S}_0(1^k)$
$(\mathcal{M}, \mathsf{st}_1) \leftarrow \mathcal{S}_1(\mathsf{st}_0)$
$\alpha \leftarrow \mathcal{S}_2^{\mathsf{TP}(\mathcal{M}, \cdot, \cdot)}(\mathsf{st}_1)$
Output: $(\mathrm{I}, \mathcal{M}, \vec{X}, \{g_\ell\}, \alpha)$

where $\{f_\ell\}$ denote the queries of $\mathcal{A}_1$ to $\mathsf{FE.Keygen}$ *and $\{g_\ell\}$ denote the functions appearing in the queries of $\mathcal{S}_2$ to* TP *such that $|\{f_\ell\}| = |\{g_\ell\}| = p$. The oracle $\mathsf{TP}(\mathcal{M}, \cdot, \cdot)$ denotes the ideal world trusted party that given the message distribution $\mathcal{M}$,* TP *first samples a message vector $\vec{X} \leftarrow M$, where $\vec{X} = \{x_{1,j}, \ldots, x_{n,j}\}_{j=1}^{q}$. It accepts input queries of the form $(g, (j_1, \ldots, j_n))$ and outputs $g(x_{1,j_1}, \ldots, x_{n,j_n})$.*