Robust Transfroming Combiners from iO to FE

Prabhanjan Ananth⋆, Aayush Jain⋆⋆, and Amit Sahai⋆⋆⋆

Abstract. Indistinguishability Obfuscation (iO) has enabled an incredible number of new and exciting applications. However, our understanding of how to actually build secure iO remains in its infancy. While many candidate constructions have been published, some have been broken, and it is unclear which of the remaining candidates are secure.

This work deals with the following basic question: *Can we hedge our bets when it comes to iO candidates?* In other words, if we have a collection of iO candidates, and we only know that at least one of them is secure, can we still make use of these candidates?

This topic was recently studied by Ananth, Jain, Naor, Sahai, and Yogev [CRYPTO 2016], who showed how to construct a robust iO combiner: Specifically, they showed that given the situation above, we can construct a single iO scheme that is secure as long as (A) at least one candidate iO scheme is a subexponentially secure iO, and (B) either the subexponential DDH or LWE assumptions hold.

In this work, we make three contributions:

– (**Better robust iO combiners.**) First, we work to improve the assumptions needed to obtain the same result as Ananth et al.: namely we show how to replace the DDH/LWE assumption with the assumption that subexponentially secure one-way functions exist.
– (**FE and NIKE from iO candidates and minimal assumptions.**) Second, we consider a broader question: what if we start with several iO candidates where only one works, but we don't care about achieving iO itself, rather we want to achieve concrete applications of iO? In this case, we are able to work with the *minimal* assumption of just polynomially secure one-way functions, and where the working iO candidate only achieves polynomial security. Under these circumstances, we show how to achieve both *functional encryption* and *non-interactive multiparty key exchance (NIKE)*.

⋆ Center for Encrypted Functionalities, Computer Science Department, UCLA. Email: `prabhanjan@cs.ucla.edu`. This work was partially supported by grant #360584 from the Simons Foundation and the grants listed under Amit Sahai.

⋆⋆ Center for Encrypted Functionalities, Computer Science Department, UCLA. Email: `aayush@cs.ucla.edu`.

⋆⋆⋆ University of California Los Angeles and Center for Encrypted Functionalities. Email: `sahai@cs.ucla.edu`. Research supported in part from a DARPA/ARL SAFEWARE award, NSF Frontier Award 1413955, NSF grants 1228984, 1136174, 1118096, and 1065276, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the ARL under Contract W911NF-15-C-0205. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

– (**Correctness Amplification for iO from polynomial security and one-way functions.**) Finally, along the way, we obtain a result of independent interest: Recently, Bitansky and Vaikuntanathan [TCC 2016] showed how to amplify the correctness of an iO scheme, but they needed subexponential security for the iO scheme and also require subexponentially secure DDH or LWE. We show how to achieve the same correctness amplification result, but requiring only polynomial security from the iO scheme, and assuming only polynomially secure one-way functions.

1 Introduction

Indistinguishability Obfuscation (iO), first defined by [4], has been a major revelation to cryptography. The discovery of the punctured programming technique by Sahai and Waters [46] has led to several interesting applications of indistinguishability obfuscation. A very incomplete list of such results includes functional encryption [23,47,2], the feasibility of succinct randomized encodings [6,12,38], time lock puzzles [7], software watermarking [15], instantiating random oracles [35] and hardness of Nash equilibrium [9,25].

On the construction side, however, iO is still at a nascent stage. The first candidate was proposed by Garg, Gentry, Halevi, Raykova, Sahai and Waters [23] from multilinear maps [22,19,28]. Since then there have many proposals of iO [3,48,28]. All these constructions are based on multilinear maps. The constructions of multilinear maps have come under scrutiny after several successful cryptanalytic attacks [36,14,16,17,18,13,43] were mounted against them. In fact, there have also been direct attacks on some of the iO candidates as well [17,43]. However, there are (fortunately) still many candidates that have survived all known cryptanalytic attacks. We refer the reader to Appendix A in [1] for a partial list of these candidates[1]. In light of this, its imperative to revisit the applications of iO and hope to weaken the trust we place on any specific known iO candidate to construct these applications.

In other words, *can we hedge our bets when it comes to iO candidates?*

If we're wrong about some candidates that seem promising right now, but not others, then can we still give explicit constructions that achieve the amazing applications of iO?

Robust iO Combiners. Recently, Ananth, Jain, Naor, Sahai and Yogev [1] considered the closely related problem of constructing an iO scheme starting from many iO candidates, such that the final iO scheme is guaranteed to be secure as long as *even one* of the iO candidates is secure. In fact, they only assume that the secure candidate satisfies correctness, and in particular, the insecure candidates could also be incorrect. This notion is termed as a *robust iO combiners* (also studied by Fischlin et al. [21] in a relaxed setting where multiple underlying iO candidates must be secure) and are useful in constructing *universal*

[1] Several recent candidates such as [41,42,24] have not been included in this list. There are currently no attacks known on these candidates as well.

iO [31] [2]. The work of [1] constructs robust iO combiners assuming the existence of a *sub-exponentially secure* iO scheme and *sub-exponentially secure DDH/ LWE*. As a consequence of this result, we can construct the above applications by combining all known iO candidates as long as one of the candidates is sub-exponentially secure.
While the work of [1] is a major advance, it leaves open two very natural questions, that we study in this work. The first question is: do we really need to assume DDH or LWE? In other words:

> *1. What assumption suffices to construct a robust iO combiner? In particular, are (sub-exponentially secure) one-way functions sufficient?*

The second, broader, question is: if we care about constructing *applications* of iO, can we do better? In particular, recent work [26] has shown that functional encryption – itself an application of iO – can be directly used to construct several applications of iO. Let us then define an *robust iO-to-FE combiner* as an object that takes several iO candidates, with the promise that at least one of them is only polynomially secure, and outputs an explicit secure functional encryption scheme. Then, let us consider the following question, which truly addresses a *minimal* assumption:

> *2. Assuming only polynomially secure one-way functions, can we construct a robust iO-to-FE combiner?*

Note that since the existence of iO does not even imply that **P**$\neq$**NP**, while functional encryption implies one-way functions, the above question lays out a minimal assumption for constructing a robust iO-to-FE combiner.

1.1 Our Contribution.

We address questions 1 and 2 in this work. We show,

Theorem 1 (Informal). *Given many iO candidates out of which at least one of them is correct and secure and additionally assuming one-way functions, we can construct a compact functional encryption scheme.*

As a corollary, we can construct an explicit functional encryption scheme assuming the *existence* of iO and one-way functions. In other words, we show that it suffices that iO exists (rather than relying on a constructive proof of it) to construct an explicit functional encryption scheme.

Corollary 1 (Informal). *Assuming polynomially secure iO and one-way functions exists, we can construct an explicit compact functional encryption scheme. In particular, the construction of functional encryption does not rely on an explicit description of the iO scheme.*

[2] A scheme Π is said to be a universal secure iO scheme if the following holds: if there exists a secure iO scheme (whose explicit description is unknown) then Π is a secure iO scheme.

Combining this result with the works of [2,10] who show how to construct iO from sub-exponentially secure compact FE, we obtain the following result.

Theorem 2 (Informal). *There exists a robust iO combiner assuming sub-exponentially secure one-way functions as long as one of the underlying iO candidates is sub-exponentially secure.*

This improves upon the result of Ananth et al. [1] who achieve the same result assuming sub-exponentially secure DDH or LWE.

Explicit NIKE from several iO candidates: Recent works of Garg and Srinivasan [27], Li and Micciancio [40], show how to achieve collusion resistant functional encryption from compact functional encryption and Garg, Pandey, Srinivasan and Zhandry [26] show how to build multi-party non interactive key exchange (NIKE) from collusion resistant functional encryption. When combined with these results, our work shows how to obtain an explicit NIKE protocol when given any one-way function, and many iO candidates with the guarantee that only one of the candidates is secure.

New Correctness Amplification Theorem for iO. En route to achieving this result, we demonstrate a new correctness amplification theorem for iO. In particular, we show how to obtain almost-correct iO starting from polynomially secure approximately-correct iO[3] and one-way functions. Prior to our work, [11] showed how to achieved the same result but starting from *sub-exponentially secure* iO and *sub-exponentially secure* DDH/ LWE.

Theorem 3 (Informal). *There is a transformation from a polynomially secure approximately-correct iO to polynomially secure almost-correct iO assuming one-way functions.*

2 Technical Overview

The goal of our work is to construct a compact functional encryption scheme starting many iO candidates out of which one of them is secure. Let us start with the more ambitious goal of building a robust compact FE combiner. If we have such a combiner, then we achieve our goal since the i^{th} compact FE candidate used in the combiner can be built from the i^{th} iO candidate using prior works [23].

To build a compact FE combiner, we view this problem via the lens of secure multi-party computation: we view every compact FE candidate as corresponding to a party in the MPC protocol; insecure candidates correspond to adversaries. Ananth et al. [1] took the same viewpoint when building an iO combiner and in particular, used *non-interactive* MPC techniques that relied on DDH/ LWE to solve this problem. Our

[3] An iO scheme is ε-approximately correct if every obfuscated circuit agrees with the original circuit on ε fraction of the inputs.

goal is however to base our combiner only on one-way functions and to achieve that, we start with an interactive MPC protocol.

A first attempt is the following: Let $\Pi_1, \ldots, \Pi_n$ be the n compact FE candidates. We start with an interactive MPC protocol for parties $P_1, \ldots, P_n$.

- To encrypt a message x, we secret share x into n additive shares. Each of these shares are encrypted using candidates $\Pi_1, \ldots, \Pi_n$.
- To generate a functional key for function f, we generate a functional key for the following function g_i using FE candidate Π_i: this function g_i takes as input message m and executes the next message function of Π_i to obtain message m'. If m' has to be sent to Π_j then it encrypts m' under the public key of Π_j and outputs the ciphertext.

The decryption algorithm proceeds as in the evaluation of the multi-party secure computation protocol. Since one of the candidates is secure, say $\mathbf{i}^{th}$ candidate, the hope is that the $\mathbf{i}^{th}$ ciphertext hides the $\mathbf{i}^{th}$ share of x and thus security of FE is guaranteed.

However, implementing the above high level idea faces the following obstacles.

Statelessness: While a party participating in a MPC protocol is stateful, the functional key is not. Hence, the next message function as part of the functional key expects to receive the previous state as input. Its not clear how to ensure without sharing state information with all the other candidates.

Randomized Functions: The functional key in the above solution encrypts a message with respect to another candidate. Since the encryption is probabilistic, we require the function for which the key is computed to be randomized. Prior works handle this problem by embedding a PRF key in the ciphertext. Since the ciphertext corresponding to i^{th} candidate could be created by j^{th} candidate, this would mean that the PRF key is shared with j^{th} candidate. This is obviously a problem if j^{th} candidate is insecure.

Oblivious Transfer: Recall that our goal was to base the combiner only on one-way functions. However, MPC requires oblivious transfer and from Impagliazzo and Rudich's result we have strong evidence to believe that oblivious transfer cannot based on one-way functions.

Correctness Amplification: A recent elegant work of Bitansky and Vaikuntanathan [11] study correctness amplification techniques in the context of indistinguishability obfuscation and functional encryption. Their correctness amplification theorems assume DDH/ LWE to achieve this result. Indeed, this work was also employed by Ananth et al. to construct an iO combiner. We need a different mechanism to handle the correctness issue if our goal is to base our construction on one-way functions.

Tackling Issues: We propose the following ideas to tackle the above issues.

Use 2-ary FE instead of compact FE: The first idea is to replace compact FE candidates with 2-ary FE[4] candidates. We can build each of 2-ary FE candidates starting from iO candidates. The advantage of using 2-ary FE is two fold:

1. It helps in addressing the issue of statelessness. The functional keys, of say i^{th} candidate, are now associated with 2-ary functions, where the first input of the function takes as input the previous state and the other input takes as input the message from another candidate. The output of this function is the updated state encrypted under the public key of the i^{th} candidate and encryption of message under public key of j^{th} candidate, where j^{th} candidate is supposed to receive this message. This way, the state corresponding to the i^{th} candidate is never revealed to any other candidate.
2. It also helps in addressing the issue of randomized functions. The first input to the function could also contain a PRF key. This key will be used to generate the randomness required to encrypt messages with respect to public keys of other candidates.

Getting Rid of OT: To deal with this issue, we use the idea of preprocessing OTs that is extensively used in the MPC literature. We preprocess all possible OTs that will be used ahead of time. While this seems to be a viable solution, this would mean that we have to preprocess as many OTs as the size of the circuit implementing the functionality. This leads to the issue of reusability: we need to make sure a set of preprocessed OTs is only used once for every evaluation of the function.

1. *Reusability*: We need to make sure a set of preprocessed OTs is only used once for every evaluation of the function.
2. *Compactness*: Since we are constructing compact FE, we need to make sure that the encryption complexity is independent of the number of pre-processed OTs.

One attempt to fix the reusability problem is to pre-process OTs during the encryption phase. This would lead to the compactness issue stated above. To resolve this, we "compress" the OTs using PRF keys. That is, to generate OTs between two parties P_i and P_j, we use a PRF key K_{ij}. The next problem is under which public key do we encrypt K_{ij}. Encrypting this under either i^{th} candidate or j^{th} candidate could compromise the key completely. The guarantee we want is that as long as one of the two candidates is honest, this key is not compromised. To solve this problem, we employ a 1-out-2 combiner of 2-ary FE combiner – given two candidates, 1-out-2 combiner is secure as long as one of them is secure. This can be achieved by computing an "onion" of two FE candidates. We refer the reader to the technical section for more details.

Correctness Amplification: [11] showed how to transform ε-approximately correct iO into an almost correct iO scheme. They do this in two steps: (i) the first step is the self reducibility step, where they transform approximately correct iO scheme into one, where the iO scheme is correct

[4] A 2-ary FE scheme is a functional encryption corresponding to 2-ary functions. A functional key of 2-ary function f decrypts two ciphertexts CT_1 (of message x) and CT_2 (of message y) to obtain $f(x, y)$.

on every input with probability close to ε, (ii) then they apply BPP amplification techniques to get almost correct iO. Their self reducibility step involves using a type of secure function evaluation scheme and they show how to construct this based on DDH and LWE. We instead show how to achieve the self reducibility step using a single key private key functional encryption scheme. The main idea is as follows: to obfuscate a circuit C, we generate a functional key of C and then obfuscate the decryption algorithm with the functional key hardwired inside it. Additionally, we give out the master secret key in the clear along with this obfuscated circuit. To evaluate on an input x, first encrypt this using the master secret key and feed this ciphertext to the obfuscated circuit, which evaluates the decryption algorithm to produce the output. This approach leads to the following issues: (i) firstly, revealing the output of the FE decryption could affect the correctness of iO: for instance, the obfuscated circuit could output $\perp$ for all inputs on which the FE decryption outputs 1, (ii) since the evaluator has the master secret key, he could feed in maliciously generated FE ciphertexts into the obfuscated circuit.
We solve (i) by using by masking the output of the circuit. Here, the mask is supplied as input to the obfuscated circuit. We solve (ii) by using NIZKs with pre-processing, a tool used by Ananth et al. to construct witness encryption combiners. This primitives can be based on one-way functions.

Our Solution in a Nutshell: Summarizing, we take the following approach to build compact FE starting from many iO candidates out of which at least one of them is correct and secure.

1. First check if the candidates are approximately correct. If not, discard the candidates.
2. Apply the new correctness amplification mechanism on all the remaining iO candidates.
3. Construct n 2-ary FE candidates from the n iO candidates obtained from the previous step.
4. Then using an onion-based approach, obtain a 2-ary FE combiner that only combines two candidates. This will lead to $N = n^2 - n$ candidates.
5. Construct a compact FE scheme starting from the above N 2-ary FE candidates and an n-party MPC protocol with OT preprocessing phase. Essentially every $(i,j)^{th}$ 2-ary FE candidate implements a channel between i^{th} and j^{th} party.

We expand on the above high level approach in the relevant technical sections.

3 Preliminaries

Let λ be the security parameter. For a distribution $\mathcal{D}$ we denote by $x \xleftarrow{\$}$ an element chosen from $\mathcal{D}$ uniformly at random. We denote that $\{\mathcal{D}_{1,\lambda}\} \approx_{c,\mu} \{\mathcal{D}_{2,\lambda}\}$, if for every PPT distinguisher $\mathcal{A}$, $\Big| Pr\big[\mathcal{A}(1^\lambda, x \xleftarrow{\$}$

$\mathcal{D}_{1,\lambda}) = 1\big] - Pr\big[\mathcal{A}(1^\lambda, x \xleftarrow{\$} \mathcal{D}_{2,\lambda}) = 1\big]\Big| \leq \mu(\lambda)$ where μ is a negligible function. For a language L associated with a relation R with denote by $(x, w) \in R$ an instance $x \in L$ with a valid witness w. For an integer $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, \ldots, n\}$. By negl we denote a negligible function. We assume that the reader is familiar with one-way functions, pseudorandom functions and in particular sub-exponential security of these primitives. We say that the one-way function is sub-exponentially secure if no polynomial time adversary inverts a random image with a probability greater than inverse sub-exponential in the length of the input.

Important Notation. We introduce some notation that will be useful throughout this work. Consider an algorithm A. We define the *time function* of A to be T if the runtime of $A(x) \leq T(|x|)$. We are only interested in time functions which satisfy the property that $T(\text{poly}(n)) = |\text{poly}(T(n))|$. In this section, we describe NIZK with Pre-Processing while in Appendix C we describe statistically binding commitments and NIWIs. We also discuss sub-exponential security of these primitives.

3.1 NIZK with Pre-Processing

We consider a specific type of zero knowledge proof system where the messages exchanged is independent of the input instance till the last round. We call this zero knowledge proof system with pre-processing. The pre-processing algorithm essentially simulates the interaction between the prover and the verifier till the last round and outputs views of the prover and the verifier.

Definition 1. *Let L be a language with relation R. A scheme* $\mathsf{PZK} = (\mathsf{PZK.Pre}, \mathsf{PZK.Prove}, \mathsf{PZK.Verify})$ *of PPT algorithms is a zero knowledge proof system with pre-processing,* PZK*, between a verifier and a prover if they satisfy the following properties. Let* $(\sigma_V, \sigma_P) \leftarrow \mathsf{PZK.Pre}(1^\lambda)$ *be a preprocessing stage where the prover and the verifier interact. Then:*

1. **Completeness***: for every $(x, w) \in R$ we have that:*

$$\Pr\left[\mathsf{PZK.Verify}(\sigma_V, x, \pi) = 1 \ : \ \pi \leftarrow \mathsf{PZK.Prove}(\sigma_P, x, w)\right] = 1.$$

   where the probability is over the internal randomness of all the PZK *algorithms.*

2. **Soundness***: for every $x \notin L$ we have that:*

$$\Pr[\exists \pi : \mathsf{PZK.Verify}(\sigma_V, x, \pi) = 1] < 2^{-n}$$

   where the probability is only over $\mathsf{PZK.Pre}$*.*

3. **Zero-Knowledge***: there exists a PPT algorithm S such that for any x, w where $V(x, w) = 1$ there exists a negligible function μ such that it holds that:*

$$\{\sigma_V, \mathsf{PZK.Prove}(\sigma_P, x, w)\} \approx_{c,\mu} \{S(x)\}$$

   We say that PZK *is sub-exponentially secure if $\mu(\lambda) = O(2^{-\lambda^c})$ for a constant $c > 0$.*

Such schemes were studied in [20,39] where they proposed constructions based on one-way functions. Sub-exponentially secure PZK can be built from sub-exponentially secure one-way functions.

4 Definitions: IO Combiner

We recall the definition of IO combiners from [1]. Suppose we have many indistinguishability obfuscation (IO) schemes, also referred to as *IO candidates*. We are additionally guaranteed that one of the candidates is secure. No guarantee is placed on the rest of the candidates and they could all be potentially broken. Indistinguishability obfuscation combiners provides a mechanism of combining all these candidates into a single monolithic IO scheme *that is secure*. We emphasize that the only guarantee we are provided is that one of the candidates is secure and in particular, it is unknown exactly which of the candidates is secure.
We formally define IO combiners next. We start by providing the syntax of an obfuscation scheme. We then present the definitions of an IO candidate and a secure IO candidate. To construct IO combiner, we need to also consider functional encryption candidates. Once we give these definitions, we present our construction in Section 6.

Syntax of Obfuscation Scheme. An obfuscation scheme associated to a class of circuits $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ with input space $\mathcal{X}_\lambda$ and output space $\mathcal{Y}_\lambda$ consists of two PPT algorithms $(\mathsf{Obf}, \mathsf{Eval})$ defined below.

- **Obfuscate,** $\overline{C} \leftarrow \mathsf{Obf}(1^\lambda, C)$: It takes as input security parameter λ, a circuit $C \in \mathcal{C}_\lambda$ and outputs an obfuscation of C, $\overline{C}$.
- **Evaluation,** $y \leftarrow \mathsf{Eval}\left(\overline{C}, x\right)$: This is usually a deterministic algorithm. But sometimes we will treat it as a randomized algorithm. It takes as input an obfuscation $\overline{C}$, input $x \in \mathcal{X}_\lambda$ and outputs $y \in \mathcal{Y}_\lambda$.

Throughout this work, we will only be concerned with *uniform* Obf algorithms. That is, Obf and Eval are represented as Turing machines (or equivalently uniform circuits).
We require that each candidate satisfy the following property called polynomial slowdown.

Definition 2 (Polynomial Slowdown.). *An obfuscation scheme $\Pi = (\mathsf{Obf}, \mathsf{Eval})$ is an IO candidate for a class of circuits $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$, with every $C \in \mathcal{C}_\lambda$ has size $\mathrm{poly}(\lambda)$, if it satisfies the following property:*
Polynomial Slowdown*: For every $C \in \mathcal{C}_\lambda$, we have the running time of Obf on input $(1^\lambda, C)$ to be $\mathrm{poly}(|C|, \lambda)$. Similarly, we have the running time of Eval on input $(\overline{C}, x)$ for $x \in \mathcal{X}_\lambda$ is $\mathrm{poly}(|\overline{C}|, \lambda)$.*

We now define various notions of correctness.

Definition 3 (Almost/ Perfect Correct IO candidate). *An obfuscation scheme $\Pi = (\mathsf{Obf}, \mathsf{Eval})$ is an almost correct IO candidate for a class of circuits $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$, with every $C \in \mathcal{C}_\lambda$ has size $\mathrm{poly}(\lambda)$, if it satisfies the following property:*

– **Almost Correctness**: *For every* $C : \mathcal{X}_\lambda \to \mathcal{Y}_\lambda \in \mathcal{C}_\lambda, x \in \mathcal{X}_\lambda$ *it holds that:*

$$\mathsf{Pr}\left[\forall x \in \mathcal{X}_\lambda, \mathsf{Eval}\left(\mathsf{Obf}(1^\lambda, C), x\right) = C(x)\right] \geq 1 - \mathsf{negl},$$

over the random coins of Obf*. The candidate is called a correct IO candidate if this probability is 1.*

Definition 4 **($\alpha-$worst-case Correctness).** *An obfuscation scheme* $\Pi = (\mathsf{Obf}, \mathsf{Eval})$ *is* $\alpha-$*worst-case correct IO candidate for a class of circuits* $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$*, with every* $C \in \mathcal{C}_\lambda$ *has size* $\mathrm{poly}(\lambda)$*, if it satisfies the following property:*

– $\alpha-$**worst-case Correctness**: *For every* $C : \mathcal{X}_\lambda \to \{0,1\} \in \mathcal{C}_\lambda, x \in \mathcal{X}_\lambda$ *it holds that:*

$$\mathsf{Pr}\left[\mathsf{Eval}\left(\mathsf{Obf}(1^\lambda, C), x\right) = C(x)\right] \geq \alpha,$$

over the random coins of Obf *and* Eval*. The candidate is correct if this probability is 1.*

Remark 1. Given any $\alpha-$worst case correct IO candidate where $\alpha > 1/2 + 1/\mathsf{poly}(\lambda)$, as observed by [11] we can gen an almost correct IO candidate while retaining security via BPP amplification.

ϵ-Secure IO candidate. If any IO candidate additionally satisfies the following (informal) security property then we define it to be a *secure* IO candidate: for every pair of circuits C_0 and C_1 that are equivalent[5] we have obfuscations of C_0 and C_1 to be indistinguishable by any PPT adversary.

Definition 5 **(ϵ-Secure IO candidate).** *An obfuscation scheme* $\Pi = (\mathsf{Obf}, \mathsf{Eval})$ *for a class of circuits* $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ *is a* ϵ*-secure IO candidate if it satisfies the following conditions:*

– **Security.** *For every PPT adversary* $\mathcal{A}$*, for every sufficiently large* $\lambda \in \mathbb{N}$*, for every* $C_0, C_1 \in \mathcal{C}_\lambda$ *with* $C_0(x) = C_1(x)$ *for every* $x \in \mathcal{X}_\lambda$ *and* $|C_0| = |C_1|$*, we have:*

$$\left|\mathsf{Pr}\left[0 \leftarrow \mathcal{A}\left(\mathsf{Obf}(1^\lambda, C_0), C_0, C_1\right)\right] - \mathsf{Pr}\left[0 \leftarrow \mathcal{A}\left(\mathsf{Obf}(1^\lambda, C_1), C_0, C_1\right)\right]\right| \leq \epsilon(\lambda)$$

Remark 2. We say that Π is a secure IO candidate if it is a ϵ-secure IO candidate with $\epsilon(\lambda) = \mathsf{negl}(\lambda)$, for some negligible function negl.

We remarked earlier that identity function is an IO candidate. However, note that the identity function is *not* a secure IO candidate. Whenever we refer an IO candidate we will specify the correctness and the security notion it satisfies. For example [4,32,23] are examples of negl-secure correct IO candidate. In particular, an IO candidate need not necessarily have any security/correctness property associated with it.
We have the necessary ingredients to define an IO combiner.

[5] Two circuits C_0 and C_1 are equivalent if they (a) have the same size, (b) have the same input domain and, (c) for every x in the input domain, $C_0(x) = C_1(x)$.

4.1 Definition of IO Combiner

We present the formal definition of IO combiner below. First, we provide the syntax of the IO combiner. Later we present the properties associated with an IO combiner.

There are two PPT algorithms associated with an IO combiner, namely, $\mathsf{CombObf}$ and $\mathsf{CombEval}$. Procedure $\mathsf{CombObf}$ takes as input circuit C along with the description of multiple correct IO candidates[6] and outputs an obfuscation of C. Procedure $\mathsf{CombEval}$ takes as input the obfuscated circuit, input x, the description of the candidates and outputs the evaluation of the obfuscated circuit on input x.

Syntax of IO Combiner. We define an IO combiner $\Pi_{\mathsf{comb}} = (\mathsf{CombObf}, \mathsf{CombEval})$ for a class of circuits $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$.

- **Combiner of Obfuscate algorithms,** $\overline{C} \leftarrow \mathsf{CombObf}(1^\lambda, C, \Pi_1, \ldots, \Pi_n)$: It takes as input security parameter λ, a circuit $C \in \mathcal{C}$, description of correct IO candidates $\{\Pi_i\}_{i \in [n]}$ and outputs an obfuscated circuit $\overline{C}$.

- **Combiner of Evaluation algorithms,** $y \leftarrow \mathsf{CombEval}(\overline{C}, x, \Pi_1, \ldots, \Pi_n)$: It takes as input obfuscated circuit $\overline{C}$, input x, descriptions of IO candidates $\{\Pi_i\}_{i \in [n]}$ and outputs y.

We define the properties associated to any IO combiner. There are three main properties – correctness, polynomial slowdown, and security. The correctness and the polynomial slowdown properties are defined on the same lines as the corresponding properties of the IO candidates.

The intuitive security notion of IO combiner says the following: suppose one of the candidates is a secure IO candidate then the output of obfuscator ($\mathsf{CombObf}$) of the IO combiner on C_0 is computationally indistinguishable from the output of the obfuscator on C_1, where C_0 and C_1 are equivalent circuits.

Definition 6 **((ϵ', ϵ)-secure IO combiner).** *Consider a circuit class $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$. We say that $\Pi_{\mathsf{comb}} = (\mathsf{CombObf}, \mathsf{CombEval})$ is a (ϵ', ϵ)-* **secure IO combiner** *if the following conditions are satisfied: Let $\Pi_1, \ldots, \Pi_n$ be n correct IO candidates for P/poly, and ϵ is a function of ϵ'.*

- **Correctness.** *Let $C \in \mathcal{C}_{\lambda \in \mathbb{N}}$ and $x \in \mathcal{X}_\lambda$. Consider the following process: (a) $\overline{C} \leftarrow \mathsf{CombObf}(1^\lambda, C, \Pi_1, \ldots, \Pi_n)$, (b) $y \leftarrow \mathsf{CombEval}(\overline{C}, x, \Pi_1, \ldots, \Pi_n)$.*
  Then with overwhelming probability over randomness of $\mathsf{CombObf}$, $\mathsf{Pr}[y = C(x)] \geq 1$, *where the probability is over $x \xleftarrow{\$} \mathcal{X}_\lambda$.*

- **Polynomial Slowdown.** *For every $C : \mathcal{X}_\lambda \to \mathcal{Y}_\lambda \in \mathcal{C}_\lambda$, we have the running time of* $\mathsf{CombObf}$ *on input $(1^\lambda, C, \Pi_1, \ldots, \Pi_n)$ to be at most* $\mathsf{poly}(|C| + n + \lambda)$. *Similarly, we have the running time of* $\mathsf{CombEval}$ *on input $(\overline{C}, x, \Pi_1, \ldots, \Pi_n)$ to be at most* $\mathsf{poly}(|\overline{C}| + n + \lambda)$.

[6] The description of an IO candidate includes the description of the obfuscation and the evaluation algorithms.

– **Security.** *Let Π_i be ϵ-secure correct IO candidate for some $i \in [n]$. For every PPT adversary $\mathcal{A}$, for every sufficiently large $\lambda \in \mathbb{N}$, for every $C_0, C_1 \in \mathcal{C}_\lambda$ with $C_0(x) = C_1(x)$ for every $x \in \mathcal{X}_\lambda$ and $|C_0| = |C_1|$, we have:*

$$\left|\mathsf{Pr}\left[0 \leftarrow \mathcal{A}\left(\overline{C_0}, C_0, C_1, \Pi_1, \ldots, \Pi_n\right)\right] - \mathsf{Pr}\left[0 \leftarrow \mathcal{A}\left(\overline{C_1}, C_0, C_1, \Pi_1, \ldots, \Pi_n\right)\right]\right| \leq \epsilon'(\lambda),$$

where $\overline{C_b} \leftarrow \mathsf{CombObf}(1^\lambda, C_b, \Pi_1, \ldots, \Pi_n)$ for $b \in \{0, 1\}$.

Some remarks are in order.

Remark 3. We say that Π_{comb} is an IO combiner if it is a (ϵ', ϵ)-secure IO combiner, where, (c) $\epsilon' = \mathsf{negl}'$ and, (d) $\epsilon = \mathsf{negl}$ with negl and negl' being negligible functions.

Remark 4. We alternatively call the IO combiner defined in Definition 6 to be a 1-out-n IO combiner. In our construction we make use of 1-out-2 IO combiner. This can be instantiated using a folklore "onion combiner" in which to obfuscate any given circuit one uses both the obfuscation algorithms to obfuscate the circuit one after the other in a nested fashion.

Remark 5. We also define robust combiner, where the syntax is the same as above except that security and correctness properties hold even if there is only one input candidate that is secure and correct. No restriction about correctness and security is placed on other candidates.

As seen in [1], a robust combiner for arbitrary many candidates imply universal obfuscation as defined below.

Definition 7 **((T, ϵ)-Universal Obfuscation).** *We say that a pair of Turing machines $\Pi_{\mathsf{univ}} = (\Pi_{\mathsf{univ}}.\mathsf{Obf}, \Pi_{\mathsf{univ}}.\mathsf{Eval})$ is a* **universal obfuscation**, *parameterized by T and ϵ, if there exists a correct ϵ-secure indistinguishability obfuscator for P/poly with time function T then Π_{univ} is an indistinguishability obfuscator for P/poly with time function* $\mathrm{poly}(T)$.

4.2 Functional Encryption

We start by providing the syntax of a functional encryption scheme.

Syntax of Functional Encryption Scheme. An functional encryption scheme associated to a class of circuits $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ consists of four polynomial time algorithms $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ defined below. Let $\mathcal{X}_\lambda$ be the message space of the scheme and $\mathcal{Y}_\lambda$ be the space of outputs for the scheme (same as the output space of $\mathcal{C}_\lambda$).

– **Setup,** $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$: It is a randomized algorithm takes as input security parameter λ and outputs a key pair $(\mathsf{MPK}, \mathsf{MSK})$

– **Encryption,** $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MPK}, m)$: It is a randomized algorithm takes the public key MPK and a message $m \in \mathcal{X}_\lambda$ and outputs an encryption of m.

- **Key Generation,** $sk_C \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, C)$: This is a randomized algorithm that takes as input the master secret key MSK and a circuit $C \in \mathcal{C}_\lambda$ and outputs a function key sk_C.
- **Decryption,** $y \leftarrow \mathsf{Dec}(sk_C, \mathsf{CT})$: This is a deterministic algorithm that takes as input the function secret key sk_C and a ciphertext CT and outputs a value $y \in \mathcal{Y}_\lambda$.

Throughout this work, we will only be concerned with *uniform* algorithms. That is, $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ are represented as Turing machines (or equivalently uniform circuits).
We require the FE scheme to satisfy the following three properties:

Correctness: For every $C : \mathcal{X}_\lambda \rightarrow \{0,1\} \in \mathcal{C}_\lambda, m \in \mathcal{X}_\lambda$ it holds that:

$$Pr\begin{bmatrix} (\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MPK}, m) \\ sk_C \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, C) \\ C(m) \leftarrow \mathsf{Dec}(sk_C, \mathsf{CT}) \end{bmatrix} \geq 1 - \mathsf{negl}(\lambda)$$

, where the probability is taken over the coins of the setup.

Compactness: Let $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$, for every $m \in \mathcal{X}_\lambda$, $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MPK}, m)$. We require that $|\mathsf{CT}| < poly(|m|, \lambda)$.

Security: We require the following informal requirement: fix any $m_0, m_1 \in \mathcal{X}_\lambda$ and $C \in \mathcal{C}_\lambda$ such that $C(m_0) = C(m_1)$, the following happens. Given the public key MPK, function secret key sk_C and cipher-text CT^* (encrypting either m_0 or m_1) it is hard to tell which is the case. Formally,

For every PPT adversary $\mathcal{A}$, for every sufficiently large $\lambda \in \mathbb{N}$, for every $C \in \mathcal{C}_\lambda$ with $C(m_0) = C(m_1)$ where $m_0, m_1 \in \mathcal{X}_\lambda$ and $|m_0| = |m_1|$, we have:

$$\left|\mathsf{Pr}\left[0 \leftarrow \mathcal{A}\Big(\mathsf{MPK}, m_0, m_1, \mathsf{CT}_0, sk_C\Big)\right] - \mathsf{Pr}\left[0 \leftarrow \mathcal{A}\Big(\mathsf{MPK}, m_0, m_1, \mathsf{CT}_1, sk_C\Big)\right]\right| \leq \epsilon(\lambda)$$

Where $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$, $\mathsf{CT}_b \leftarrow \mathsf{Enc}(\mathsf{MPK}, m_b)$ for $b \in \{0,1\}$ and $sk_C \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, C)$ and probability is taken over the coins of all the algorithms. We say that the FE scheme is secure if ϵ is a negligible function. Further we say that it is sub-exponentially secure if $\epsilon = O(2^{-\lambda^c})$ for some constant $c > 0$.

4.3 Definition of 2-ary Functional Encryption Candidate

We now define 2-ary (public-key) functional encryption candidates, also referred to as MIFE *candidates*). We start by providing the syntax of a MIFE scheme.

Syntax of 2-ary Functional Encryption Scheme. A MIFE scheme associated to a class of circuits $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ consists of four polynomial time algorithms $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ defined below. Let $\mathcal{X}_\lambda$ be the message space of the scheme and $\mathcal{Y}_\lambda$ be the space of outputs for the scheme (same as the output space of $\mathcal{C}_\lambda$).

- **Setup,** $(\mathsf{EK}_1, \mathsf{EK}_2, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$: It is a randomized algorithm takes as input security parameter λ and outputs a keys $(\mathsf{EK}_1, \mathsf{EK}_2, \mathsf{MSK})$. Here EK_1 and EK_2 are encryption keys for indices 1 and 2 and MSK is the master secret key.

- **Encryption,** $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{EK}_i, m)$: It is a randomized algorithm takes the encryption key EK_i for any index $i \in [2]$ and a message $m \in \mathcal{X}_\lambda$ and outputs an encryption of m (encrypted under EK_i).

- **Key Generation,** $sk_C \leftarrow \mathsf{KeyGen}\,(\mathsf{MSK}, C)$: This is a randomized algorithm that takes as input the master secret key MSK and a 2-input circuit $C \in \mathcal{C}_\lambda$ and outputs a function key sk_C.

- **Decryption,** $y \leftarrow \mathsf{Dec}\,(sk_C, \mathsf{CT}_1, \mathsf{CT}_2)$: This is a deterministic algorithm that takes as input the function secret key sk_C and a ciphertexts CT_1 and CT_2 (encrypted under EK_1 and EK_2 respectively). Then it outputs a value $y \in \mathcal{Y}_\lambda$.

Throughout this work, we will only be concerned with *uniform* algorithms. That is, $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ are represented as Turing machines (or equivalently uniform circuits).

We define the notion of an MIFE candidate below. The following definition of multi-input functional encryption scheme incorporates only the correctness and compactness properties of a multi-input functional encryption scheme [30]. In particular, an MIFE candidate need not necessarily have any security property associated with it. Formally,

Definition 8 (Correct MIFE candidate). *A multi-input functional encryption scheme* $\mathsf{MIFE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ *is a correct MIFE candidate for a class of circuits* $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$*, with every* $C \in \mathcal{C}_\lambda$ *has size* $\mathrm{poly}(\lambda)$*, if it satisfies the following properties:*

- **Correctness***: For every* $C : \mathcal{X}_\lambda \times \mathcal{X}_\lambda \to \{0,1\} \in \mathcal{C}_\lambda, m_1, m_2 \in \mathcal{X}_\lambda$ *it holds that:*

$$Pr\begin{bmatrix} (\mathsf{EK}_1, \mathsf{EK}_2, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{CT}_i \leftarrow \mathsf{Enc}(\mathsf{EK}_i, m_i)\ i \in [2] \\ sk_C \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, C) \\ C(m_1, m_2) \leftarrow \mathsf{Dec}(sk_C, \mathsf{CT}_1, \mathsf{CT}_2) \end{bmatrix} \geq 1 - \mathsf{negl}(\lambda)$$

  , where negl *is a negligible function and the probability is taken over the coins of the setup only.*

- **Compactness***: Let* $(\mathsf{EK}_1, \mathsf{EK}_2, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$*, for every* $m \in \mathcal{X}_\lambda$ *and* $i \in [2]$*,* $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{EK}_i\, m)$*. We require that* $|\mathsf{CT}| < poly(|m|, \lambda)$*.*

A scheme is an MIFE candidate if it only satisfies the correctness and compactness property.

Selective Security. We recall indistinguishability-based selective security for MIFE. This security notion is modeled as a game between a challenger C and an adversary $\mathcal{A}$ where the adversary can request for functional keys and ciphertexts from C. Specifically, $\mathcal{A}$ can submit 2-ary function queries f and respond with the corresponding functional keys. It submits message queries of the form (m_1^0, m_2^0) and (m_1^1, m_2^1) and receive encryptions of messages m_i^b for $i \in [2]$, and for some random bit $b \in \{0,1\}$. The adversary $\mathcal{A}$ wins the game if she can guess b with probability significantly more than 1/2 if the following properties are satisfied:

- $f(m_1^0, \cdot)$ is functionally equivalent to $f(m_1^1, \cdot)$.
- $f(\cdot, m_2^0)$ is functionally equivalent to $f(\cdot, m_2^1)$
- $f(m_1^0, m_2^0) = f(m_1^1, m_2^1)$

Formal definition is presented next.

ϵ-Secure MIFE candidate. If any MIFE candidate additionally satisfies the following (informal) security property then we define it to be a *secure* MIFE candidate:

Definition 9 (ϵ-Secure MIFE candidate). *A scheme* MIFE *for a class of circuits* $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ *and message space* $\mathcal{X}_\lambda$ *is a ϵ-secure FE candidate if it satisfies the following conditions:*

- MIFE *is a correct and compact MIFE candidate with respect to* $\mathcal{C}$,
- **Security.** *For every PPT adversary* $\mathcal{A}$, *for every sufficiently large* $\lambda \in \mathbb{N}$, *we have:*

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

where the probability is taken over coins of all algorithms. For each $b \in B$ *and* $\lambda \in \mathbb{N}$, *the experiment* $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{MIFE}}(1^\lambda, b)$ *is defined below:*

1. **Challenge message queries:** $\mathcal{A}$ *outputs* (m_1^0, m_2^0) *and* (m_1^0, m_2^0) *where each* $m_j^i \in \mathcal{X}_\lambda$
2. *The challenger computes* $\mathsf{Setup}(1^\lambda) \rightarrow (\mathsf{EK}_1, \mathsf{EK}_2, \mathsf{MSK})$. *It then computes* $\mathsf{CT}_1 \leftarrow \mathsf{Enc}(\mathsf{EK}_1, m_1^b)$ *and* $\mathsf{CT}_2 \leftarrow \mathsf{Enc}(\mathsf{EK}_1, m_2^b)$. *Challenger hands* $\mathsf{CT}_1, \mathsf{CT}_2$ *to the adversary.*
3. $\mathcal{A}$ *submits functions* f_i *to the challenger satisfying the constraint given below.*
   - $f_i(m_1^0, \cdot)$ *is functionally equivalent to* $f_i(m_1^1, \cdot)$.
   - $f_i(\cdot, m_2^0)$ *is functionally equivalent to* $f_i(\cdot, m_2^1)$
   - $f_i(m_1^0, m_2^0) = f_i(m_1^1, m_2^1)$

   For every i, *the adversary gets* $sk_{f_i} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, f_i)$.
4. *Adversary submits the guess* b'. *The output of the game is* b'.

Remark 6. We say that MIFE is a secure MIFE candidate if it is a ϵ-secure FE candidate with $\epsilon(\lambda) = \mathsf{negl}(\lambda)$, for some negligible function negl.

5 MPC Framework

We consider a MPC framework in the pre-processing model described below. Intuitively the input is pre-processed and split amongst n deterministic parties which are also given some correlated randomness. Then, they run a protocol together to compute $f(x)$ for any function f of the input x. The syntax consist of with the following algorithms:

- $\mathsf{Preproc}(1^\lambda, n, x) \rightarrow (x_1, \mathsf{corr}_1, .., x_n, \mathsf{corr}_n)$: This algorithm takes as input $x \in \mathcal{X}_\lambda$, the number of parties computing the protocol n, and the input λ. It outputs strings x_i, corr_i for $i \in [n]$. Each $\mathsf{corr}_i = \mathsf{corr}_i(r)$ is represented both a function and a value depending on the context. $(x_1, .., x_n)$ forms a secret sharing of x.
- $\mathsf{Eval}(\mathsf{Party}_1(x_1, \mathsf{corr}_1), .., \mathsf{Party}_n(x_n, \mathsf{corr}_n), f) \rightarrow f(x)$: The evaluate algorithm is a protocol run by n parties with Party_i having input x_i, corr_i. Each Party_i is deterministic. The algorithm also takes as input the function $f \in \mathcal{C}_\lambda$ of size bounded by $\mathsf{poly}(\lambda)$ and it outputs $f(x)$.

We now list the notations used for the protocol.

1. The number of rounds in the protocol is given by a polynomial $t_f(\lambda, n, |x|)$.
2. For every $i \in [n]$, $\mathsf{corr}_i = \{\mathsf{corr}_{i,j}\}_{j \neq i}$. Let $len_f = len_f(\lambda, n)$ denote a polynomial. Then, for each $i, j \in [n]$ such that $i \neq j$, $\mathsf{corr}_{i,j}$ and $\mathsf{corr}_{j,i}$ are generated as follows. Sample $r_{i,j} \xleftarrow{\$} \{0,1\}^{len_f}$ then compute $\mathsf{corr}_{i,j} = \mathsf{corr}_{i,j}(r_{i,j})$ and $\mathsf{corr}_{j,i} = \mathsf{corr}_{j,i}(r_{i,j})$.
3. There exists an efficiently computable function ϕ_f that takes as input a round number $k \in [t_f]$ and outputs $\phi_f(k) = (i, j)$. Here, i represents that the sender of the message at k^{th} round is Party_i and the recipient is Party_j.
4. The efficiently computable next message function for every round $k \in [t_f]$, M_k does the following. Let $\phi_f(k) = (i, j)$. Then, M_k takes as input $(x_i, y_1, .., y_{k-1}, \mathsf{corr}_{i,j})$ and outputs the next message as y_k.

Correctness. : We require the following correctness property to be satisfied by the protocol. For every $n, \lambda \in \mathbb{N}$, $x \in \mathcal{X}_\lambda$, $f \in \mathcal{C}_\lambda$ it holds that:

$$Pr\left[\begin{array}{c}(x_1, \mathsf{corr}_1, .., x_n, \mathsf{corr}_n) \leftarrow \mathsf{Preproc}(1^\lambda, n, x) \\ \mathsf{Eval}(\mathsf{Party}_1(x_1, \mathsf{corr}_1), ..., \mathsf{Party}_n(x_n, \mathsf{corr}_n), f) \rightarrow f(x)\end{array}\right] = 1$$

, Here the probability is taken over coins of the algorithm $\mathsf{Preproc}$.

Security Requirement. We require the security against static corruption of $n-1$ semi-honest parties. Informally the security requirement is the following. There exists a polynomial time algorithm that takes as input $f(x)$ and inputs of $n-1$ corrupt parties $\{(\mathsf{corr}_i, x_i)\}_{i \neq i^*}$ and simulates the outgoing messages of Party_{i^*}. Formally, Consider a PPT adversary $\mathcal{A}$. Let the associated PPT simulator be Sim. We define the security experiment below.

$\underline{\mathsf{Expt}_{real,\mathcal{A}}(1^\lambda)}$

- $\mathcal{A}$ on input 1^λ outputs n, the circuit f and input x along with the index of the honest party, $i^* \in [n]$.
- Secret share x into $(x_1, .., x_n)$.
- Part of the pre-processing step is performed by the adversary. For every $i > j$ such that $i^* \neq i$ and $i^* \neq j$, $\mathcal{A}$ samples $r_{i,j} \xleftarrow{\$} \{0,1\}^{len_f}$. Then, it computes, $\mathsf{corr}_{i,j} = \mathsf{corr}_{i,j}(r_{i,j})$ and $\mathsf{corr}_{j,i} = \mathsf{corr}_{j,i}(r_{i,j})$.

- Sample $r_j \xleftarrow{\$} \{0,1\}^{len_f}$ for $j \neq i^*$. Then compute $\mathsf{corr}_{i^*,j} = \mathsf{corr}_{i^*,j}(r_j)$ and $\mathsf{corr}_{j,i^*} = \mathsf{corr}_{j,i^*}(r_j)$. We denote $\mathsf{corr}_i = \{\mathsf{corr}_{i,j}\}_{j \neq i}$. This completes the pre-processing step.
- Let $y_1, .., y_{t_f}$ be the messages computed by the parties in the protocol computing $f(x)$. Output $\left(\{x_i, \mathsf{corr}_i\}_{i \neq i^*}, y_1, .., y_k\right)$. In MPC literature $(\{x_i, \mathsf{corr}_i\}_{i \neq i^*}, y_1, .., y_k)$ is referred to the view of the adversary in this experiment. We refer this as $\mathsf{view}_{\mathsf{Expt}_{real,\mathcal{A}}}$.

$\underline{\mathsf{Expt}_{ideal,\mathcal{A}}(1^\lambda)}$
- $\mathcal{A}$ on input 1^λ outputs n, the circuit f and input x along with the index of the honest party, $i^* \in [n]$.
- Secret share x into $(x_1, .., x_n)$.
- Part of the pre-processing step is performed by the adversary. For every $i > j$ such that $i^* \neq i$ and $i^* \neq j$, $\mathcal{A}$ samples $r_{i,j} \xleftarrow{\$} \{0,1\}^{len_f}$. Then, it computes, $\mathsf{corr}_{i,j} = \mathsf{corr}_{i,j}(r_{i,j})$ and $\mathsf{corr}_{j,i} = \mathsf{corr}_{j,i}(r_{i,j})$.
- Sample $r_j \xleftarrow{\$} \{0,1\}^{len_f}$ for $j \neq i^*$. Then compute $\mathsf{corr}_{i^*,j} = \mathsf{corr}_{i^*,j}(r_j)$ and $\mathsf{corr}_{j,i^*} = \mathsf{corr}_{j,i^*}(r_j)$. This completes the pre-processing step.
- Compute $Sim\left(1^\lambda, 1^{|f|}, f(x), \{x_i, \mathsf{corr}_i\}_{i \neq i^*}\right)$. Output the result. We refer this as $\mathsf{view}_{\mathsf{Expt}_{ideal,\mathcal{A}}}$.

We require that the output of both the above experiments is computationally indistinguishable from each other. That is,

Definition 10 (Security). *Consider a PPT adversary $\mathcal{A}$ and let the associated PPT simulator be Sim. For every PPT distinguisher $\mathcal{D}$, for sufficiently large security parameter λ, it holds that:*

$$\left|\mathsf{Pr}\left[1 \leftarrow \mathcal{D}\left(\mathsf{Expt}_{real,\mathcal{A}}(1^\lambda)\right)\right] - \mathsf{Pr}\left[1 \leftarrow \mathcal{D}(\mathsf{Expt}_{ideal,\mathcal{A}}\left(1^\lambda\right))\right]\right| \leq \mathsf{negl}(\lambda),$$

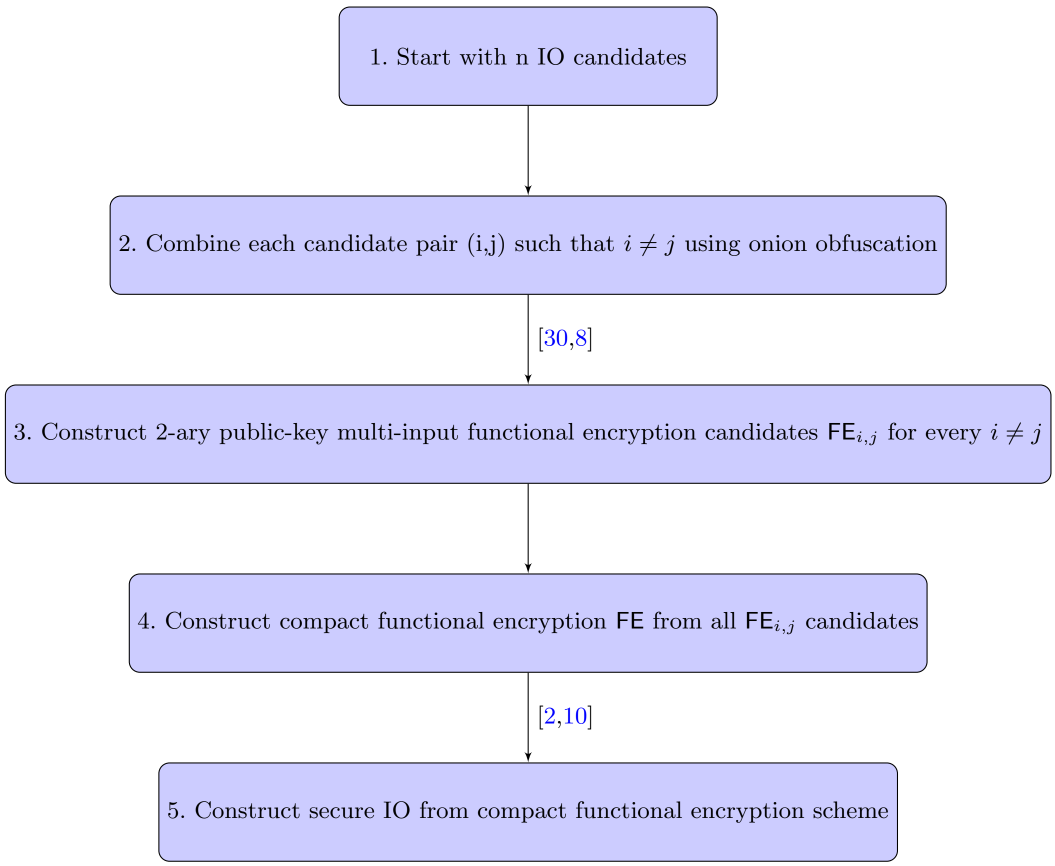
where negl *is some negligible function.*

Instantiation of MPC Framework: We show how to instantiate this MPC framework. We use a 1-out-of-n (i.e., $n-1$ of them are insecure) information theoretically secure MPC protocol secure against passive adversaries [29,37] in the OT hybrid model. We then replace the OT oracle by preprocessing all the OTs [5] before the execution of the protocol begins. Note that every OT pair is associated exactly with a pair of parties.

6 Construction of IO Combiner

In this section, we describe our construction. First we sketch the roadmap. We start with n IO candidates, $\Pi_1, .., \Pi_n$ and construct $n^2 - n$ IO candidates $\Pi_{i,j}$ where $i \neq j$. $\Pi_{i,j}$ is constructed by using an onion obfuscation combiner. Each candidate $\Pi_{i,j}$ is now used to construct a 2-ary public-key multi-input functional encryption scheme $\mathsf{FE}_{i,j}$ candidates using [30,8](This step uses an existence of one-way function). This is because [30] uses an existence of a public-key encryption, statistically binding commitments and statistically sound non-interactive witness-indistinguishable proofs. All these primitives can be constructed using IO and one-way functions as shown in works such as [46,8]. These primitives maintain binding/soundness as long as the underlying candidate is correct.

Any candidate $\mathsf{FE}_{i,j}$ is secure as long as either Π_i or Π_j is secure. This follows from the security of onion obfuscation combiner. We describe below how to construct a compact functional encryption FE from these multi-input functional encryption candidates in Section 6.1. Finally, using [2,10] and relying on complexity leveraging we construct a secure IO candidate Π_{comb} from FE. Below is a flowchart describing the roadmap.



6.1 Constructing compact FE from n IO candidates

Consider the circuit class $\mathcal{C}$. We now present our construction for a compact functional encryption scheme FE for $\mathcal{C}$ starting from compact multi-input functional encryption candidates $\mathsf{FE}_{i,j}$ for $\mathcal{C}$. Let Γ be a secure

MPC protocol described in Section 5. Let λ be the security parameter and F denote a pseudorandom function (PRF). Denote the PRF (indexed by the output length) as $F_{len} : \{0,1\}^{\lambda} \times \{0,1\}^{*} \rightarrow \{0,1\}^{len(\lambda)}$ where len is a polynomial. Finally let Com be a statistically binding commitment scheme.

<u>$\mathsf{FE.Setup}(1^{\lambda})$</u>

1. **Setting up MIFE candidates:**
   - For every $i, j \in [n]$ and $i \neq j$ run $\mathsf{FE}_{i,j}.\mathsf{Setup}(1^{\lambda}) \rightarrow (\mathsf{EK}_{i,j,1}, \mathsf{EK}_{i,j,2}, \mathsf{MSK}_{i,j})$
2. **Sample NIWI prover strings**
   - Run $\mathsf{NIWI}_{i,j}.\mathsf{Setup} \rightarrow \sigma_{i,j}$ for $i, j \in [n]$ and $i \neq j$. Recall, $\mathsf{NIWI}_{i,j}$ is a non-interactive statistically sound witness-indistinguishable proof scheme (in the CRS model) constructed using IO candidate $\Pi_{i,j}$ and any one-way function as done in [8][7]. This proof scheme remains sound if the underlying obfuscation candidate is correct/almost correct. The proof retains witness indistinguishability if the candidate is additionally secure.
   - Output $\mathsf{MPK} = \{\mathsf{EK}_{i,j,2}, \sigma_{i,j}\}_{i,j \in [n], i \neq j}$ and $\mathsf{MSK} = \{\mathsf{MSK}\}_{i,j \in [n], i \neq j}$.

<u>$\mathsf{FE.Enc}(\mathsf{MPK}, m)$</u>

1. **MPC Preprocessing**
   - Run $\mathsf{Preproc}(1^{\lambda}, n, m) \rightarrow (m_1, \mathsf{corr}'_1, .., m_n, \mathsf{corr}'_n)$. Compute commitments $Z_{\mathsf{in},i} = \mathsf{Com}(m_i)$ for all $i \in [n]$. Let $r_{\mathsf{in},i}$ be the corresponding randomness.
2. **Sample and commit PRF keys**
   - Sample PRF keys $K_{i,j}$ for $i, j \in [n]$ and $i \neq j$ with the constraint that $K_{i,j} = K_{j,i}$. Compute $Z_{i,j} = \mathsf{Com}(K_{i,j})$ for $i, j \in [n]$ and $i \neq j$. Let $r_{i,j}$ be the corresponding randomness.
   - Sample PRF keys $K^{'}_{i,j}$ for $i, j \in [n]$ such that $i \neq j$.
3. **Compute encryptions**
   - For every $i, j \in [n]$ and $i \neq j$ compute $\mathsf{CT}_{i,j} = \mathsf{FE}_{i,j}.\mathsf{Enc}(\mathsf{EK}_{i,j,1}, m_i, K_{i,j}, K^{'}_{i,j}, \{Z_{\mathsf{in},k}, Z_{k,j}\}_{k,j \in [n], k \neq j}, r_{i,j}, r_{\mathsf{in},i}, \perp)$. Here $\perp$ is a slot of size $\mathsf{poly}(\lambda)$, which is described later.
   - Output $\mathsf{CT} = \{\mathsf{CT}_{i,j}\}_{i \neq j}$

<u>$\mathsf{FE.KeyGen}(\mathsf{MSK}, C)$</u>

Let t_C denote the number of rounds for the MPC protocol Γ for computing the circuit C. Let len_{msg} denote the maximum length of any message sent in the protocol while computing C on input.

1. **Computing commitments**
   - Compute $Z_{\mathsf{out},i} \leftarrow \mathsf{Com}(\perp^{len_{msg}})$ for $i \in [t_C]$.
2. **Compute secret-key encryptions**
   - Let E by a secret-key encryption scheme. Run $\mathsf{E.Setup}(1^{\lambda}) \rightarrow sk$. For every $i \in [t_C]$, compute $c_i = \mathsf{E.Enc}(sk, 0)$. These encryptions encrypt messages of sufficient length (described later).
3. **Generate keys**
   - Sample a random $\mathsf{tag} \in \{0,1\}^{\lambda}$
   - For every round $k \in [t_C]$, let $\phi(k) = (i', j')$, generate a key $sk_{C,k} \leftarrow \mathsf{FE}_{i',j'}.\mathsf{KeyGen}(\mathsf{MSK}_{i',j'}, G_k)$ where G_k is described in Figure 1.

$$G_k\left[\{\sigma_{i,j}\}_{i,j\in[n].i\neq j}, c_k, \{Z_{\mathsf{out},i}\}_{i\in[t_C]}, \mathsf{tag}\right]$$

Hardwired values: NIWI verifier strings $\{\sigma_i\}_{i\in[n]}$, Encryption c_k, Commitments $\{Z_{\mathsf{out},i}\}_{i\in[t_C]}$, a random tag

Inputs 1: $m_{i'}, K_{i',j'}, K'_{i',j'}, \{Z_{\mathsf{in},i}, Z_{i,j}\}_{i,j\in[n],i\neq j}, r_{i',j'}, r_{\mathsf{in},i'}, s$

Inputs 2: $x_1, \pi_1, ..., x_{k-1}, \pi_{k-1}$

- Initialise $\mathsf{counter} = 0$. For every $\mathsf{counter} \leq k-1$ do the following:
  - $\phi(\mathsf{counter}) = (i,j)$. Check that $\mathsf{NIWI}_{i,j}.\mathsf{Verify}(\sigma_{i,j}, x_1, .., x_{\mathsf{counter}}, \pi_{\mathsf{counter}}) = 1$ for the language L_i described below.
  - Output $\perp$ if any check fails.
- If $s \neq \perp$. Outputs $\mathsf{E.Dec}(s, c_k)$.
- Otherwise, let $\phi(k) = (i',j')$. Compute $\mathsf{corr}_{i',j'} = \mathsf{corr}_{i',j'}(F(K_{i',j'}, \mathsf{tag}))$. Compute $x_k = M_k(m_{i'}, x_1, .., x_{k-1}, \mathsf{corr}_{i',j'})$.
- Compute a $\mathsf{NIWI}_{i',j'}$ proof π_k using the reference string $\sigma_{i',j'}$ and randomness $F(K'_{i',j'}, x_1, .., x_{k-1}, x_k)$ that the statement $(x_1, .., x_k) \in L_{i'}$ (language defined below). This proof is computed using the witness as $m_{i'}, K_{i',j'}, r_{i',j'}, r_{\mathsf{in},i'}, 0$ where $\phi(k) = (i',j')$.
- Output (x_k, π_k)

<u>**Language L_i:**</u> An NP language L_i is defined by the relation as defined below:

Instance: $x_1, .., x_n$ where $n \in [t_C]$

Witness: $m_i, K_{i,j}, r_{i,j}, r_{\mathsf{in},i}, r_n$

- Let $\phi(n) = (i,j)$. If not, output 0.
- Let $\mathsf{corr}_{i,j} = \mathsf{corr}_{i,j}(F(K_{i,j}, \mathsf{tag}))$. Check that either, $x_n = M_n(m_i, x_1, .., x_{n-1}, \mathsf{corr}_{i,j})$, $Z_{i,j} = \mathsf{Com}(K_{i,j}; r_{i,j})$ and $Z_{\mathsf{in},i} = \mathsf{Com}(m_i; r_{\mathsf{in},i})$. Or,
- $Z_{\mathsf{out},n} = \mathsf{Com}(y_n; r_n)$, where $y_n \neq \perp$ and $y_n = x_n$. If the check passes output 1 otherwise 0.

Fig. 1: Circuit G_k

<u>$\mathsf{FE.Dec}(sk_C, \mathsf{CT})$</u>

1. **Evaluating the MPC protocol for circuit C**
   - For every round $n \in [t_C]$, compute x_n, π_n iteratively from $x_1, \pi_1, .., x_{n-1}, \pi_{n-1}$ as described below.
     a Compute $\phi(n) = (i,j)$. Then, compute $\mathsf{CT}_n = \mathsf{FE}_{i,j}.\mathsf{Enc}(\mathsf{EK}_{i,j,2}, x_1, \pi_1, .., x_{n-1}, \pi_{n-1})$.
     b Run $(x_n, \pi_n) \leftarrow \mathsf{FE}_{i,j}.\mathsf{Dec}(sk_{C,n}, \mathsf{CT}_{i,j}, \mathsf{CT}_n)$
     c Set $x_0 = \pi_0 = \perp$
   - Output x_{t_C}

Correctness: If the underlying MPC protocol is correct and the multi-input functional encryption candidates $\mathsf{FE}_{i,j}$ are correct then one can inspect that our scheme satisfies correctness.

[7] We note that we could have also used NIZKs with pre-processing based on one-way functions. The construction becomes a little complicated with that.

Compactness. Compactness is discussed next. The cipher-text encrypting any message m, consists of $\mathsf{FE}_{i,j}$ encryptions $\mathsf{CT}_{i,j}$ for any $i, j \in [n]$ such that $i \neq j$. Each $\mathsf{CT}_{i,j}$ encrypts $m_i, K_{i,j}, K^{'}_{i,j}, \{Z_{\mathsf{in},k}, Z_{k,j}\}_{k,j\in[n],k\neq j}$, $r_{i,j}, r_{\mathsf{in},i}, \perp$. Note that m_i is of the same length of the message where as $K_{i,j}, K^{'}_{i,j}$ are just the PRF keys that are of length λ. $\{Z_{\mathsf{in},k}, Z_{k,j}\}_{k,j\in[n],k\neq j}$ are commitments of m_i and the PRF keys respectively while $r_{i,j}$ and $r_{\mathsf{in},i}$ is the randomness used for the commitments $Z_{i,j}$ and $Z_{\mathsf{in},i}$. $\perp$ is a slot of size $\mathsf{poly}(\lambda)$ (which is the length of the decryption key for scheme E). All these strings are of a fixed polynomial size (polynomial in $n, \lambda, |m|$). If the underlying scheme $\mathsf{FE}_{i,j}$ is compact, the scheme FE is also compact. The completed proof of the following theorem can be found in Appendix B. We give a brief sketch here.

Theorem 4. *Consider the circuit class $\mathcal{C} = P/poly$. Assuming Γ is a secure MPC protocol for $\mathcal{C}$ according to the framework described in Section 5 and one-way functions then scheme* FE *is a secure functional encryption scheme as long as there is $i^* \in [n]$ such that Π_{i^*} is a secure candidate.*

Proof (Sketch). We now sketch the security proof of this theorem. Let Π_{i^*} is a secure IO candidate. This implies $\mathsf{FE}_{i^*,j}$ and FE_{j,i^*} is secure for any $j \neq i^*$. We use this crucially in our proofs. We employ the standard hybrid argument to prove the theorem. In the first hybrid (Hyb_1), the message M_b is encrypted honestly with $b \xleftarrow{\$} \{0, 1\}$. In the final hybrid ($\mathsf{Hyb}_9$), the ciphertext contains no information about b. At this point, the probability of guessing the bit b is exactly $1/2$. By arguing indistinguishability of every consecutive intermediate hybrids, we show that the probability of guessing b in the first hybrid is negligibly close to $1/2$ (or the advantage is 0), which proves the theorem.
The first hybrid corresponds to the regular FE security game. Then we switch to a hybrid where the secret-key encryption cipher-text c_i in the function keys for all rounds $i \in [t_C]$ are hard-wired as encryptions of the output of the MIFE decryption in those rounds. This can be done, because the cipher-text and the function key fixes these outputs (as a function of PRF keys, e.t.c). Then, we change the commitments $Z_{\mathsf{out},n}$ to commtiments of message output in round n (for n such that the i^* is the receiving or sending party in that round). This security holds due to the security of the commitment. In the next hybrid, we rely on the security of the scheme $\mathsf{FE}_{i^*,j}$ and FE_{j,i^*} by generating encryptions that does not contain the PRF keys and the openings of the commitments but only contain the secret key for the encryption scheme E. Now we invoke the security of the PRF to generate proofs π_n hard-wired in c_n for any round n (for n such that the i^* is the receiving or sending party in that round) randomly. Next, we rely on the security of $\mathsf{NIWI}_{i^*,j}$ and NIWI_{j,i^*} to use the opening of $Z_{\mathsf{out},n}$ (for n such that the i^* is the receiving or sending party in that round) to generate the proofs. Now relying on the security of commitment scheme, we make the commitments $Z_{i^*,j}, Z_{j,i^*}$ and $Z_{\in,i^*}$ to commit to $\perp$. Then we use the security of the PRF to generate $\mathsf{corr}_{i^*,j}$ and corr_{j,i^*} (used for generating outputs x_n) randomly. Finally, we invoke the security of the MPC framework (by using the simulator) to make the game independent of b.

6.2 Summing Up: Combiner Construction

We now give the combiner construction:

– $\mathsf{CombObf}(1^\lambda, C, \Pi_1, .., \Pi_n)$: Use $\Pi_1, .., \Pi_n$ and any one-way function to construct a compact functional encryption FE as in Section 6. Use [2,10] to construct an obfuscator Π_{comb}. Output $\overline{C} \leftarrow \Pi_{comb}(1^\lambda, C)$.
– $\mathsf{CombEval}(\overline{C}, x)$: Output $\Pi_{comb}.\mathsf{Eval}(\overline{C}, x)$.

Correctness of the scheme is straight-forward to see because of the correctness of FE as shown in Section 6. The security follows from the sub-exponential security of construction in Section 6. The construction in Section 6 is sub-exponentially secure as long as the underlying primitives are sub-exponentially secure.
We now state the theorem.

Theorem 5. *Assuming sub-exponentially secure one-way functions, the construction described above is a* $(\mathsf{negl}, 2^{-\lambda^c})-$*secure IO combiner for P/poly where* $c > 0$ *is a constant and* negl *is some negligible function.*

7 From Combiner to Robust Combiner

The combiner described in section 6, is not robust. It guarantees no security/correctness if the underlying candidates are not correct. A robust combiner provides security/correctness as long as there exists one candidate Π_{i^*} such that it is secure and correct. There is no other restriction placed on the other set of candidates. A robust combiner for arbitrary many candidates imply universal obfuscation [1].
In this section we describe how to construct a robust combiner. The idea is the following.

- We just correct the candidates (upto overwhelming probability) before feeding it as input to the combiner.
- First, we leverage the fact that secure candidate is correct. We transform each candidate so that all candidates are $(1 - 1/\lambda)-$worst case correct while maintaining security of the secure candidate.
- Then using [11] we convert a worst-case correct candidate to an almost correct candidate.

In the discussion below, we assume $\mathcal{C}$ consists polynomial size circuits with one bit output. One can construct obfuscator for circuits with multiple output bits from obfuscator with one output bit. For simplicity let us assume that $\mathcal{C}_\lambda$ consists of circuits with input length $p(\lambda)$ for some polynomial p.

7.1 Generalised Secure Function Evaluation

The starting point to get a worst-case correct IO candidate is a variant of “Secure Function Evaluation” (SFE) scheme as considered in [11]. They use SFE to achieve worst-case correctness by obfuscating evaluation function of SFE for the desired circuit C. To evaluate on input x, the evaluator first encodes x according to the SFE scheme and feeds it as an input to the obfuscated program. Then, it finally decodes the result as the

output of the obfuscated program. Worst case correctness is guaranteed because using the information hard-wired in the obfuscated program its hard to distinguish an encoding of any input x_1 from that of x_2.
We essentially use the same idea except that we consider a variant of SFE with a setup algorithm (which produces secret parameters), and the evaluation function for the circuit C is not public. It requires some helper information to perform evaluation on the input encodings.
We consider a generalised variant of secure function evaluation [11] with the following properties. Let $\mathcal{C}_\lambda$ be the the allowed set of circuits. Let $\mathcal{X}_\lambda$ and $\mathcal{Y}_\lambda$ denote the ensemble of inputs and outputs. A secure function evaluation scheme consists of following algorithms:

- $\mathsf{Setup}(1^\lambda)$: On Input 1^λ, the setup algorithm outputs secret parameters SP.
- $\mathsf{CEncode}(\mathsf{SP}, C)$: The randomized circuit encoding algorithm on input a circuit $C \in \mathcal{C}_\lambda$ and SP outputs another $\tilde{C} \in \mathcal{C}_\lambda$.
- $\mathsf{InpEncode}(\mathsf{SP}, x)$: The randomized input encoding algorithm on input $x \in \mathcal{X}_\lambda$ and SP outputs $(\tilde{x}, z) \in \mathcal{X}_\lambda \times \mathcal{Z}_\lambda$.
- $\mathsf{Decode}(y, z)$: Let $y = \tilde{C}(\tilde{x})$. The deterministic decoding algorithm takes as input y and z to recover $C(x) \in \mathcal{Y}_\lambda$.

We require the following properties:

Input Secrecy: For any $x_1, x_2 \in \mathcal{X}_\lambda$ and any circuit $C \in \mathcal{C}_\lambda$, it holds that:

$$\{\mathsf{CEncode}(\mathsf{SP}, C), \mathsf{InpEncode}(\mathsf{SP}, x_1) | \mathsf{SP} \leftarrow \mathsf{Setup}(1^\lambda)\} \approx_c$$

$$\{\mathsf{CEncode}(\mathsf{SP}, C), \mathsf{InpEncode}(\mathsf{SP}, x_2) | \mathsf{SP} \leftarrow \mathsf{Setup}(1^\lambda)\}$$

Correctness: For any circuit $C \in \mathcal{C}_\lambda$ and any input x, it holds that:

$$Pr[\mathsf{Decode}(\tilde{C}(\tilde{x}), z) = C(x)] = 1$$

Where $\mathsf{SP} \leftarrow \mathsf{Setup}(1^\lambda), \tilde{C} \leftarrow \mathsf{CEncode}(\mathsf{SP}, C), (\tilde{x}, z) \leftarrow \mathsf{InpEncode}(\mathsf{SP}, x)$ and the probability is taken over coins of all the algorithms.

Functionality: For any equivalent circuits C_0, C_1, $\mathsf{SP} \leftarrow \mathsf{Setup}(1^\lambda)$, $\tilde{C}_0 \leftarrow \mathsf{CEncode}(\mathsf{SP}, C_0)$ and $\tilde{C}_1 \leftarrow \mathsf{CEncode}(\mathsf{SP}, C_1)$, it holds that $\tilde{C}_0$ is equivalent to $\tilde{C}_1$ with overwhelming probability over the coins of setup and the circuit encoding algorithm. This captures the behaviour of the circuit encodings when evaluated on maliciously generated input encodings.

7.2 Modified Obfuscation Candidate

In this section we achieve the following. Given any candidate Π, we transform it to a candidate Π' such that the following holds:

- If Π is both secure and correct, then so is Π'.
- Otherwise Π' is guaranteed to be $(1 - 1/\lambda)-$worst-case correct.

In either, case we can amplify its correctness to get an almost correct IO candidate, which can be used by our combiner construction. Given any IO candidate Π we now describe a modified IO candidate Π'. For simplicity let us assume that $\mathcal{C}_\lambda$ consists of circuits with one bit output and input space $\mathcal{X}_\lambda$ corresponds to the set $\{0,1\}^{p(\lambda)}$ for some polynomial p. Let SFE be a secure function evaluation scheme as described in Section 7.1 for $\mathcal{C}_\lambda$ with $\mathcal{Z}_\lambda = \mathcal{Y}_\lambda = \{0,1\}$.

- **Obfuscate:** On input the security parameter 1^λ and $C \in \mathcal{C}_\lambda$, first run $\mathsf{SP} \leftarrow \mathsf{SFE.Setup}(1^\lambda)$, compute $\tilde{C} \leftarrow \mathsf{CEncode}(\mathsf{SP}, C)$. We now define an algorithm $\mathsf{Obf}_{int,\Pi}$ that takes as input $\tilde{C}$ and 1^λ and does the following:
  - Compute $\overline{C} \leftarrow \Pi.\mathsf{Obf}(1^\lambda, \tilde{C})$.
  - Then sample randomly $x_1, .., x_{\lambda^2} \in \{0,1\}^{p(\lambda)}$. Compute $(\tilde{x}_i, z_i) \leftarrow \mathsf{InpEncode}(\mathsf{SP}, x_i)$. Check that $\Pi.\mathsf{Eval}(\overline{C}, \tilde{x}_i) = \tilde{C}(\tilde{x}_i)$ for all $i \in [\lambda^2]$.
  - If the check passes output $\overline{C}$, otherwise output $\tilde{C}$ [8].

  Output of the obfuscate algorithm is $(\mathsf{SP}, \mathsf{Obf}_{int,\Pi}(\tilde{C}))$.
- **Evaluate:** On input $(\mathsf{SP}, \overline{C})$ and an input x, first compute $(\tilde{x}, z) \leftarrow \mathsf{InpEncode}(\mathsf{SP}, x)$. Then compute $\tilde{y} \leftarrow \Pi.\mathsf{Eval}(\overline{C}, \tilde{x})$ or $\tilde{y} \leftarrow \overline{C}(\tilde{x})$ depending on the case if $\overline{C} = \tilde{C}$ or not. We define as an intermediate evaluate algorithm, i.e. $\tilde{y} = \mathsf{Eval}_{int,\Pi}(\overline{C}, \tilde{x})$.

  Output $y = \mathsf{SFE.Decode}(\tilde{y}, z)$.

Few claims are in order:

Theorem 6. *Assuming* SFE *is a secure function evaluation scheme as described in section 7.1, if Π is a secure and correct candidate, then so is, Π'.*

Proof. We deal with this one by one. First we argue security. Note that when Π is correct, the check at λ^2 random points passes. In this case the obfuscation algorithm always outputs $(\mathsf{SP}, \Pi.\mathsf{Obf}(1^\lambda, \tilde{C}_b))$ where $\tilde{C}_b \leftarrow \mathsf{SFE.CEncode}(\mathsf{SP}, C_b)$ for $b \in \{0,1\}$ and $\mathsf{SP} \leftarrow \mathsf{SFE.Setup}(1^\lambda)$. Since, $\tilde{C}_0$ is equivalent to $\tilde{C}_1$ due to functionality property of the SFE scheme, the security holds due to the security of Π.

The correctness holds due to the correctness of SFE and Π.

Theorem 7. *Assuming* SFE *is a secure function evaluation scheme as described in section 7.1, if Π is an IO candidate, then Π' is $(1-2/\lambda)$-worst case correct IO candidate.*

Proof. The check step in the obfuscate algorithm ensures the following: Using chernoff bound it follows that, with overwhelming probability, for any circuit C,

$$Pr[\Pi'.\mathsf{Eval}(\mathsf{SP}, \overline{C}, x) = C(x) | (\mathsf{SP}, \overline{C}) \leftarrow \Pi'.\mathsf{Obf}(1^\lambda, C), x \xleftarrow{\$} \mathcal{U}_{p(\lambda)}] \geq (1 - 1/\lambda) \tag{1}$$

[8] This step ensures circuit-specific correctness. Note that any correct candidate will always pass the step. Any candidate that is not correct with high enough probability will not pass the check. In this case, the algorithm outputs the circuit in the clear.

We now prove that for any x_1, x_2 it holds that, with overwhelming probability over coins of obfuscate algorithm, for any circuit C,

$$\begin{aligned}|Pr[\mathit{\Pi}'.\mathsf{Eval}(\mathsf{SP}, \overline{C}, x_1) = C(x_1)|(\mathsf{SP}, \overline{C}) \leftarrow \mathit{\Pi}'.\mathsf{Obf}(1^\lambda, C)]-\\ Pr[\mathit{\Pi}'.\mathsf{Eval}(\mathsf{SP}, \overline{C}, x_2) = C(x_2)|(\mathsf{SP}, \overline{C}) \leftarrow \mathit{\Pi}'.\mathsf{Obf}(1^\lambda, C)]| \leq \mathsf{negl}(\lambda)\end{aligned} \tag{2}$$

This is because for any input x,

$$\begin{aligned}&Pr[\mathit{\Pi}'.\mathsf{Eval}(\mathsf{SP}, \overline{C}, x) = C(x)|(\mathsf{SP}, \overline{C}) \leftarrow \mathit{\Pi}'.\mathsf{Obf}(1^\lambda, C)] =\\ &\quad Pr[\mathsf{Eval}_{int,\mathit{\Pi}}(\overline{C}, \tilde{x}) = \tilde{C}(\tilde{x})|\mathsf{SP} \leftarrow \mathsf{Setup}(1^\lambda), \tilde{C} \leftarrow \mathsf{CEncode}(\mathsf{SP}, C),\\ &\quad (\tilde{x}, z) \leftarrow \mathsf{InpEncode}(\mathsf{SP}, x), \overline{C} \leftarrow \mathsf{Obf}_{int,\mathit{\Pi}}(1^\lambda, \tilde{C})]\end{aligned} \tag{3}$$

Note that due to the input secrecy property of the SFE scheme we have that,

$$\begin{aligned}|Pr[\mathsf{Eval}_{int,\mathit{\Pi}}(\overline{C}, \tilde{x}_1) = \tilde{C}(\tilde{x_1})|\mathsf{SP} \leftarrow \mathsf{Setup}(1^\lambda), \tilde{C} \leftarrow \mathsf{CEncode}(\mathsf{SP}, C),\\ (\tilde{x_1}, z_1) \leftarrow \mathsf{InpEncode}(\mathsf{SP}, x_1), \overline{C} \leftarrow \mathsf{Obf}_{int,\mathit{\Pi}}(1^\lambda, \tilde{C})]-\\ Pr[\mathsf{Eval}_{int,\mathit{\Pi}}(\overline{C}, \tilde{x}_2) = \tilde{C}(\tilde{x_2})|\mathsf{SP} \leftarrow \mathsf{Setup}(1^\lambda), \tilde{C} \leftarrow \mathsf{CEncode}(\mathsf{SP}, C),\\ (\tilde{x_2}, z_2) \leftarrow \mathsf{InpEncode}(\mathsf{SP}, x_2), \overline{C} \leftarrow \mathsf{Obf}_{int,\mathit{\Pi}}(1^\lambda, \tilde{C})]| < \mathsf{negl}(\lambda)\end{aligned} \tag{4}$$

for a negligible function negl. Otherwise we can build a reduction $\mathcal{R}$ that given any circuit-encoding, input encoding pair $\tilde{C}, \tilde{x}_b$ decides if $b = 0$ or $b = 1$ with a non-negligible probability. The reduction just computes $\overline{C} \leftarrow \mathsf{Obf}_{int,\mathit{\Pi}}(\tilde{C})$ and checks if $\mathsf{Eval}_{int,\mathit{\Pi}}(\overline{C}, \tilde{x}_b) = \tilde{C}(\tilde{x}_b)$.

Using the pigeon-hole principle and Equation 1, for any $C \in \mathcal{C}_\lambda$ there exists x^* such that,

$$|Pr[\mathit{\Pi}'.\mathsf{Eval}(\mathsf{SP}, \overline{C}, x^*) = C(x^*)|(\mathsf{SP}, \overline{C}) \leftarrow \mathit{\Pi}'.\mathsf{Obf}(1^\lambda, C)] \geq (1 - 1/\lambda) \tag{5}$$

Now substituting $x_1 = x$ and $x_2 = x^*$ in Equation 4 and then plugging into Equation 3 gives us,

$$\begin{aligned}|Pr[\mathit{\Pi}'.\mathsf{Eval}(\mathsf{SP}, \overline{C}, x) = C(x)|(\mathsf{SP}, \overline{C}) \leftarrow \mathit{\Pi}'.\mathsf{Obf}(1^\lambda, C)]-\\ Pr[\mathit{\Pi}'.\mathsf{Eval}(\mathsf{SP}, \overline{C}, x^*) = C(x^*)|(\mathsf{SP}, \overline{C}) \leftarrow \mathit{\Pi}'.\mathsf{Obf}(1^\lambda, C)]| \leq \mathsf{negl}(\lambda)\end{aligned} \tag{6}$$

Substituting result of Equation 5 gives us the desired result. That is, For any circuit C and input x, it holds that,

$$\begin{aligned}|Pr[\mathit{\Pi}'.\mathsf{Eval}(\mathsf{SP}, \overline{C}, x) = C(x)|(\mathsf{SP}, \overline{C}) \leftarrow \mathit{\Pi}'.\mathsf{Obf}(1^\lambda, C)] \geq &(1 - 1/\lambda) - \mathsf{negl}(\lambda)\\ > &(1 - 2/\lambda)\end{aligned} \tag{7}$$

This proves the result.

7.3 Instantiation of SFE

To instantiate SFE as described in Section 7.1, we use any single-key functional encryption scheme. To compute the circuit encoding for any circuit C we compute a function key for a circuit that has hard-wired a function key sk_{H_C} for a circuit H_C that takes as input (x, b) and outputs $C(x) \oplus b$. This circuit uses the hard-wired function key to decrypt the input. To encode the input x, we just compute an FE encryption of (x, b). But this does not suffice, because then for any equivalent circuits C_0 and C_1, the circuit encodings are not equivalent. Hence, we use a one-time zero-knowledge proof system to prove that the cipher-text is consistent. The details follow next. Our decoding/evaluation operation is not randomized in contrast to [11], hence this allows us to directly argue security with polynomial loss, instead of going input by input.

Theorem 8. *Assuming (non-compact) public-key functional encryption scheme for a single function key query exists, there exists an* SFE *according to the definition in Section 7.1.*

Proof. Let FE denote any public-key functional encryption scheme for a single function key. Let PZK denote a non-interactive zero knowledge proof system with pre-processing. We now describe the scheme.

- $\mathsf{Setup}(1^\lambda)$: The setup takes as input the security parameter 1^λ. It first runs $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$ and $\mathsf{PZK.Pre}(1^\lambda) \rightarrow (\sigma_P, \sigma_V)$. Output $\mathsf{SP} = (\mathsf{MPK}, \mathsf{MSK}, \sigma_P, \sigma_V)$.
- $\mathsf{CEncode}(\mathsf{SP}, C)$**:** The algorithm on input (SP, C) does the following.
  - Compute $sk_{H_C} \leftarrow \mathsf{FE.KeyGen}(\mathsf{MSK}, H_C)$. H_C represents a circuit that on input (x, b) outputs $C(x) \oplus b$.
  - Let H be the circuit described in figure 2. Output H.
- $\mathsf{InpEncode}(\mathsf{SP}, x)$**:** On input an SP and an input x do the following:
  - Sample a random bit b and compute $\mathsf{CT} = \mathsf{FE.Enc}(\mathsf{MPK}, x, b; r_1)$.
  - Compute the NIZK with pre-processing proof π proving that $\mathsf{CT} \in L$ using the witness (x, b, r_1).
  - Output $((\mathsf{CT}, \pi), b) = (\tilde{x}, b)$
- $\mathsf{Decode}(y, b)$**:** Output $y \oplus b$.

We now discuss the properties:

Correctness. It is straightforward to see correctness as it follows from the completeness of the proof system and correctness of the functional encryption scheme.

Functionality: Let H_b denote an circuit encoding of circuit C_b for $b \in \{0, 1\}$ where C_0 and C_1 are equivalent circuits. The circuit takes as input (CT, π) where π is a proof that CT is an encryption of (x, b') for some x and a bit b'. Then it verifies the proof and decrypts the cipher-text using a function key that computes $C_b(x) \oplus b'$. Since, the proof system is statistically sound and FE scheme is correct this property is satisfied with overwhelming probability over the coins of the setup.

Input Secrecy: We want to show that for any circuit C and inputs x_0, x_1:

$$\{(H, \tilde{x}_0) | H \leftarrow \mathsf{CEncode}(\mathsf{SP}, C), \mathsf{SP} \leftarrow \mathsf{Setup}(1^\lambda), (\tilde{x}_0, z) \leftarrow \mathsf{InpEncode}(\mathsf{SP}, x_0)\} \approx_c$$

$H\,[\sigma_V, \mathsf{MPK}, sk_{H_C}]$

Hardwired values: NIZK Verifier sting σ_V, Functional encryption key MPK, Function secret key sk_{H_C}
Inputs: FE ciphertext CT, Proof π

- First run $\mathsf{PZK.Verify}(\sigma_V, \mathsf{CT}, \pi)$. The verification is with respect to the NP language L defined below. If it fails output $\perp$.
- Otherwise, output $\mathsf{FE.Dec}(sk_{H_C}, \mathsf{CT})$

Language L: An NP language L is defined by the relation as defined below:
Instance: A functional encryption key MPK and an FE ciphertext CT
Witness: x, b, r

- $\mathsf{CT} = \mathsf{FE.Enc}(\mathsf{MPK}, x, b; r)$ where $b \in \{0, 1\}$.

Fig. 2: Circuit H

$$\{(H, \tilde{x}_1) | H \leftarrow \mathsf{CEncode}(\mathsf{SP}, C), \mathsf{SP} \leftarrow \mathsf{Setup}(1^\lambda), (\tilde{x}_1, z) \leftarrow \mathsf{InpEncode}(\mathsf{SP}, x_1)\}$$

We claim this in a number of hybrids. The first one corresponds to the actual game where $\tilde{x}_0$ is given while the last one corresponds to the case of $\tilde{x}_1$. We also show that the hybrids are indistinguishable.

Hyb_0 : This hybrid corresponds to the following experiment for C, x_0. To run setup we run $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$. Then we sample a random bit b and compute $\mathsf{CT} \leftarrow \mathsf{FE.Enc}(\mathsf{MPK}, x_0, b)$. We sample $(\sigma_P, \sigma_V) \leftarrow \mathsf{PZK.Pre}(1^\lambda)$ and compute a proof π using PZK prover string σ_P and a witness of $\mathsf{CT} \in L$. We output $\mathsf{SP} = (\mathsf{MPK}, \mathsf{MSK}, \sigma_P, \sigma_V)$. This SP is used to encode C by computing a functional encryption key for circuit H_C first (sk_{H_C}). Call this circuit H (this circuit depends upon, σ_V and sk_{H_C}).

Hyb_1 : This hybrid is the same as the previous one except that we generate π, σ_V differently. We run the simulator of PZK system and compute $(\sigma_V, \pi) \leftarrow Sim(\mathsf{CT})$.

Hyb_2 : This hybrid is the same as the previous one except that we generate CT differently. $\mathsf{CT} = \mathsf{FE.Enc}(\mathsf{MPK}, x_1, b \oplus C(x_0) \oplus C(x_1))$.

Hyb_3 : This hybrid is the same as the previous one except that (σ_P, σ_V) are generated honestly.

Hyb_4 : This hybrid is the same as the previous one except that CT is generated as $\mathsf{FE.Enc}(\mathsf{MPK}, x_1, b)$. this corresponds to the experiment for input x_1.

We make a note of the following claims.

- Hyb_0 is indistinguishable to Hyb_1 due to the zero-knowledge security of PZK proof.
- Hyb_1 is indistinguishable to Hyb_2 due to the security of FE proof.
- Hyb_2 is indistinguishable to Hyb_3 due to the zero-knowledge security of PZK.
- Hyb_3 is indistinguishable to Hyb_4 are identical as b is a random bit.

Corollary 2. *Assuming public-key encryption exists[45], there exists an SFE scheme satisfying requirements described in Section 7.1.*

Remark 7. We note that such a scheme can be instantiated from one-way functions alone. The idea is to use a secret-key functional encryption for single function query along with a statistically binding commitment scheme. The public parameters now include a commitment of a master secret key which is used to proof consistency of the cipher-text. Since IO and one-way functions itself imply public-key encryption [46] we do not describe this construction

7.4 Robust Combiner: Construction

We now describe our robust combiner. On input the candidates $\Pi_1, .., \Pi_n$, we transform them using the SFE scheme as done in Section 7.2 so that they are $(1 - 1/\lambda)-$worst-case correct. Then using majority trick as in [11], we convert them to almost correct. Plugging it to the construction in Section 6, gives us the desired result. Finally, we also state our theorem about universal obfuscation

Theorem 9. *Assuming sub-exponentially secure one-way functions, there exists a* $(\mathsf{poly}, \epsilon)$*-Universal Obfuscation with* $\epsilon = O(2^{-\lambda^c})$ *for any constant* $c > 0$ *and any polynomial* poly.

References

1. Ananth, P., Jain, A., Naor, M., Sahai, A., Yogev, E.: Universal constructions and robust combiners for indistinguishability obfuscation and witness encryption. In: CRYPTO (2016)
2. Ananth, P., Jain, A.: Indistinguishability obfuscation from compact functional encryption. In: CRYPTO (2015)
3. Barak, B., Garg, S., Kalai, Y.T., Paneth, O., Sahai, A.: Protecting obfuscation against algebraic attacks. In: EUROCRYPT (2014)
4. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. In: CRYPTO (2001)
5. Beaver, D.: Precomputing oblivious transfer. In: CRYPTO (1995)
6. Bitansky, N., Garg, S., Lin, H., Pass, R., Telang, S.: Succinct randomized encodings and their applications. In: STOC (2015)
7. Bitansky, N., Goldwasser, S., Jain, A., Paneth, O., Vaikuntanathan, V., Waters, B.: Time-lock puzzles from randomized encodings. In: ITCS (2016)

8. Bitansky, N., Paneth, O.: Zaps and non-interactive witness indistinguishability from indistinguishability obfuscation. In: TCC (2015)
9. Bitansky, N., Paneth, O., Rosen, A.: On the cryptographic hardness of finding a nash equilibrium. In: FOCS (2015)
10. Bitansky, N., Vaikuntanathan, V.: Indistinguishability obfuscation from functional encryption. In: FOCS (2015)
11. Bitansky, N., Vaikuntanathan, V.: Indistinguishability obfuscation: From approximate to exact. In: TCC (2016)
12. Canetti, R., Holmgren, J., Jain, A., Vaikuntanathan, V.: Indistinguishability obfuscation of iterated circuits and RAM programs. In: STOC (2015)
13. Cheon, J.H., Fouque, P., Lee, C., Minaud, B., Ryu, H.: Cryptanalysis of the new CLT multilinear map over the integers. In: EUROCRYPT (2016)
14. Cheon, J.H., Han, K., Lee, C., Ryu, H., Stehlé, D.: Cryptanalysis of the multilinear map over the integers. In: EUROCRYPT (2015)
15. Cohen, A., Holmgren, J., Nishimaki, R., Vaikuntanathan, V., Wichs, D.: Watermarking cryptographic capabilities. In: STOC (2016)
16. Coron, J., Gentry, C., Halevi, S., Lepoint, T., Maji, H.K., Miles, E., Raykova, M., Sahai, A., Tibouchi, M.: Zeroizing without low-level zeroes: New MMAP attacks and their limitations. In: CRYPTO (2015)
17. Coron, J.S., Gentry, C., Halevi, S., Lepoint, T., Maji, H.K., Miles, E., Raykova, M., Sahai, A., Tibouchi, M.: Zeroizing without low-level zeroes: New mmap attacks and their limitations. In: Advances in Cryptology–CRYPTO 2015, pp. 247–266. Springer (2015)
18. Coron, J., Lee, M.S., Lepoint, T., Tibouchi, M.: Cryptanalysis of GGH15 multilinear maps. In: CRYPTO (2016)
19. Coron, J., Lepoint, T., Tibouchi, M.: Practical multilinear maps over the integers. In: CRYPTO (2013)
20. De-Santis, A., Micali, S., Persiano, G.: Non-interactive zero-knowledge with preprocessing. In: CRYPTO (1988)
21. Fischlin, M., Herzberg, A., Noon, H.B., Shulman, H.: Obfuscation combiners. Cryptology ePrint Archive, Report 2016/289 (2016), http://eprint.iacr.org/
22. Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. In: Johansson, T., Nguyen, P.Q. (eds.) Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7881, pp. 1–17. Springer (2013), http://dx.doi.org/10.1007/978-3-642-38348-9_1
23. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: FOCS (2013)
24. Garg, S., Miles, E., Mukherjee, P., Sahai, A., Srinivasan, A., Zhandry, M.: Secure obfuscation in a weak multilinear map model. In: TCC 2016-B
25. Garg, S., Pandey, O., Srinivasan, A.: Revisiting the cryptographic hardness of finding a nash equilibrium. In: CRYPTO (2016)

26. Garg, S., Pandey, O., Srinivasan, A., Zhandry, M.: Breaking the sub-exponential barrier in obfustopia. IACR Cryptology ePrint Archive 2016 (2016)
27. Garg, S., Srinivasan, A.: Single-key to multi-key functional encryption with polynomial loss. In: TCC 2016-B
28. Gentry, C., Gorbunov, S., Halevi, S.: Graph-induced multilinear maps from lattices. In: TCC (2015)
29. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: STOC (1987)
30. Goldwasser, S., Gordon, S.D., Goyal, V., Jain, A., Katz, J., Liu, F., Sahai, A., Shi, E., Zhou, H.: Multi-input functional encryption. In: EUROCRYPT (2014)
31. Goldwasser, S., Kalai, Y.T.: Cryptographic assumptions: A position paper. In: TCC (2016)
32. Goldwasser, S., Rothblum, G.N.: On best-possible obfuscation. In: TCC (2007)
33. Groth, J., Ostrovsky, R., Sahai, A.: Non-interactive zaps and new techniques for NIZK. In: CRYPTO (2006)
34. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM J. Comput. (1999)
35. Hohenberger, S., Sahai, A., Waters, B.: Replacing a random oracle: Full domain hash from indistinguishability obfuscation. In: EUROCRYPT (2014)
36. Hu, Y., Jia, H.: Cryptanalysis of GGH map. In: EUROCRYPT (2016)
37. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer - efficiently. In: CRYPTO (2008)
38. Koppula, V., Lewko, A.B., Waters, B.: Indistinguishability obfuscation for turing machines with unbounded memory. In: STOC (2015)
39. Lapidot, D., Shamir, A.: Publicly verifiable non-interactive zero-knowledge proofs. In: CRYPTO (1990)
40. Li, B., Micciancio, D.: Compactness vs collusion resistance in functional encryption. In: TCC 2016-B
41. Lin, H.: Indistinguishability obfuscation from constant-degree graded encoding schemes. In: EUROCRYPT (2016)
42. Lin, H., Vaikunthanathan, V.: Indistinguishability obfuscation from ddh-like assumptions on constant-degree graded encodings. In: FOCS (2016)
43. Miles, E., Sahai, A., Zhandry, M.: Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over ggh13. Cryptology ePrint Archive, Report 2016/147 (2016), http://eprint.iacr.org/
44. Naor, M.: Bit commitment using pseudorandomness. Journal of Cryptology (1991)
45. Sahai, A., Seyalioglu, H.: Worry-free encryption: functional encryption with public keys. In: Proceedings of the 17th ACM conference on Computer and communications security. pp. 463–472. ACM (2010)
46. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: STOC (2014)

47. Waters, B.: A punctured programming approach to adaptively secure functional encryption. Cryptology ePrint Archive, Report 2014/588 (2014)
48. Zimmerman, J.: How to obfuscate programs directly. In: EUROCRYPT (2015)

A Constructing FE combiner from LWE

We now summarize how we can combine n many FE candidates to construct a single candidate that is secure as long as one of the candidate is both secure and correct. The idea is to use a Threshold Homomorphic Encryption scheme (TFHE). In a TFHE scheme, there is one public key but n secret keys. As in any FHE scheme, the scheme allows for evaluation of ciphertexts. Each secret key allows one to produce a randomized partial decryptions. These partial decryptions can then be "added" to recover the decrypted value. The security guarantee is two fold: First, given any ciphertext CT along with $n-1$ secret keys and an underlying message, its possible to statistically simulate the partial decryption done by the missing key. Second, revealing any $n-1$ secret keys, reveal nothing about the ciphertext.

The scheme is described as follows: The setup consists of sampling n key pairs corresponding to underlying FE candidates. To encrypt, the encrypter first samples a TFHE public key and secret keys for n parties and encrypts the message under its public key. Then, under candidate i, it encrypts the TFHE ciphertext along with partial secret key i (along with some other information like a PRF key e.t.c.). The function key for function f, corresponds to generating keys for the function under each candidate i. Each key computes a partial decryption of the ciphertext evaluated for f using TFHE secret key i. The security goes by "puncturing" out secret key corresponding to i^* where i^* is the secure candidate. This is similar to what is done in [1] for IO.

For robustness, one needs more work, this can be done by an approach similar to one described in for IO[1].

B Security Proof of FE construction

Theorem 10. *Consider a circuit class $\mathcal{C} = P/poly$. Assuming Γ is a secure MPC protocol for $\mathcal{C}$ according to the framework described in Section 5 and one-way functions then scheme* FE *is a secure functional encryption scheme as long as there is $i^* \in [n]$ such that Π_{i^*} is a secure candidate.*

Proof. Let Π_{i^*} is a secure IO candidate. Assuming this implies $\mathsf{FE}_{i^*,j}$ and FE_{j,i^*} is secure for any $j \neq i^*$. We use this crucially in our proofs. We employ the standard hybrid argument to prove the theorem. In the first hybrid (Hyb_1), the message M_b is encrypted honestly with $b \xleftarrow{\$} \{0,1\}$.

In the final hybrid (Hyb_9), the ciphertext contains no information about b. At this point, the probability of guessing the bit b is exactly $1/2$. By arguing indistinguishability of every consecutive intermediate hybrids, we show that the probability of guessing b in the first hybrid is negligibly close to $1/2$ (or the advantage is 0), which proves the theorem.

In the following set of hybrids, denote $\mathcal{A}$ to be a PPT adversary.

$\underline{\mathsf{Hyb}_1}$: In this hybrid, the adversary $\mathcal{A}$ submits (M_0, M_1) to the challenger. The challenger sets $m = M_b$ and runs the setup for the FE scheme honestly to get $(\mathsf{MPK}, \mathsf{MSK})$. Then he computes $\mathsf{CT} = \mathsf{FE.Enc}(\mathsf{MPK}, m)$ and sends $(\mathsf{MPK}, \mathsf{CT})$ to $\mathcal{A}$. $\mathcal{A}$ then requests for a key for circuit C such that $C(M_0) = C(M_1)$. Challenger responds by giving out an honestly generated key $sk_C \leftarrow \mathsf{FE.KeyGen}(\mathsf{MSK}, C)$. $\mathcal{A}$ now submits his guess b'. Output of Hyb_1 is b'.

$\underline{\mathsf{Hyb}_2}$: This hybrid is similar to the previous hybrid except that function key sk_C for the circuit C is generated differently. Note that once we fix m, the PRF keys $\{K_{i,j}\}_{i \neq j}$, the input shares $\{m_i\}_{i \in [n]}$, and the random tag used in the key, all outputs $\{x_i\}_{i \in [t_C]}$ of the MPC protocol while computing $C(m)$ gets fixed. This is because fixing the keys $K_{i,j}$ fixes $\mathsf{corr}_{i,j} = \mathsf{corr}_{i,j}(F(K_{i,j}, \mathsf{tag}))$ and $\mathsf{corr}_{j,i} = \mathsf{corr}_{j,i}(F(K_{i,j}, \mathsf{tag}))$. By π_n we denote the $\mathsf{NIWI}_{i,j}$ proof of the statement $(x_1, .., x_n) \in L_i$, computed using the prover string $\sigma_{i,j}$ using the randomness $F(K^{'}_{i,j}, x_1, .., x_n)$ and the witness described in the program G_k in Figure 1. The corresponding proof π_n also gets fixed as it uses $F(K^{'}_{i,j}, x_1, ..., x_n)$ as the randomness and it is a function of $x_1, .., x_{n-1}$ (along with some fixed commitments and their openings).
For any round $n \in [t_C]$ such that, $\phi(n) = (i^*, j)$ or (j, i^*) for any $j \neq i^*$, we do the following. Compute $c_n = \mathsf{E.Enc}(sk, x_n, \pi_n)$, where sk is the sampled secret-key for the encryption scheme E used while generating function key. For all other rounds n, generate c_n as before as encryption of 0. Use these ciphertexts to generate the function keys $sk_{C,n}$.

Informally, the above two hybrids are indistinguishable due to the security of the encryption scheme E.

$\underline{\mathsf{Hyb}_3}$: This hybrid is similar to the previous hybrid except that function key sk_C for the circuit C is generated differently. The only difference is the way the commitments $\{Z_{\mathsf{out},i}\}_{i \in [t_C]}$ used in generating function keys are generated differently. For any round $n \in t_C$ such that $\phi(n) = (i^*, j)$ or (j, i^*), $Z_{\mathsf{out},n}$ is generated as commitment of x_n. For other rounds n, $Z_{\mathsf{out},n}$ is generated like before as a commitment of $\perp$.

Informally, the above two hybrids are indistinguishable due to the hiding property of Com.

$\underline{\mathsf{Hyb}_4}$: This hybrid is similar to the previous hybrid except that the ciphertext CT is generated differently. In particular, for all $j \neq i^*$, $\mathsf{CT}_{i^*,j}$ and CT_{j,i^*} are generated differently. Compute $\mathsf{CT}_{i^*,j} = \mathsf{FE}_{i^*,j}.\mathsf{Enc}(\mathsf{EK}_{i^*,j,1}, m_{i^*}, \perp, \perp, \{Z_{\mathsf{in},k}, Z_{k,i}\}_{k,i\in[n],k\neq i}, \perp, \perp, sk)$ and $\mathsf{CT}_{j,i^*} = \mathsf{FE}_{j,i^*}.\mathsf{Enc}(\mathsf{EK}_{j,i^*,1}, m_j, \perp, \perp, \{Z_{\mathsf{in},k}, Z_{k,i}\}_{k,i\in[n],k\neq i}, \perp, \perp, sk)$.

Informally, the above two hybrids are indistinguishable due to the security of the functional encryption schemes $\{\mathsf{FE}_{i^*,j}, \mathsf{FE}_{j,i^*}\}_{j\neq i^*}$

$\underline{\mathsf{Hyb}_5}$: This hybrid is similar to the previous hybrid except that the function key sk_C for the circuit C is generated differently. In particular, for any round $n \in [t_C]$ such that $\phi(n) = (i^*, j)$ or (j, i^*) for some j, compute c_n as an encryption of x_n, π_n. Here, the proof π_n is generated as before but instead of using $F(K'_{i^*,j}, x_1, .., x_n)$ as randomness, use a random string to generate the proof.

Informally, the above two hybrids are indistinguishable due to the security of the PRF.

$\underline{\mathsf{Hyb}_6}$: This hybrid is similar to the previous hybrid except that the function key sk_C for the circuit C is generated differently. In particular, for any round $n \in [t_C]$ such that $\phi(n) = (i^*, j)$ or (j, i^*), for some j, compute c_n as an encryption of x_n, π_n. Here, the proof π_n is generated using the trapdoor witness (i.e. opening of $Z_{\mathsf{out},n}$).

Informally, the above two hybrids are indistinguishable due to the witness indistinguishability property of $\{\mathsf{NIWI}_{i^*,j}, \mathsf{NIWI}_{j,i^*}\}_{j\neq i^*}$.

$\underline{\mathsf{Hyb}_7}$: This hybrid is similar to the previous hybrid except that the ciphertext CT is generated differently. In particular, for all $j \neq i^*$, $Z_{i^*,j}$ and Z_{j,i^*} are generated as commitments of $\perp$. Z_{in,i^*} is also made to be a commitment of $\perp$.

Informally, the above two hybrids are indistinguishable because of the hiding property of Com.

$\underline{\mathsf{Hyb}_8}$: This hybrid is similar to the previous hybrid except that the ciphertext CT is generated differently. In particular, for any j, the randomness for generating $\mathsf{corr}_{i^*,j}$ and corr_{j,i^*} is uniformly chosen (instead of being derived as $F(K_{i^*,j}, \mathsf{tag})$). This is used to compute the messages x_i for any round i, such that $\phi(i) = (i^*, j)$ or $\phi(i) = (j, i^*)$ for some $j \neq i^*$.

Informally, the above two hybrids are indistinguishable due to the security of the PRF.

$\underline{\mathsf{Hyb}_9}$: This hybrid is similar to the previous hybrid except that the ciphertext CT is generated differently. In particular, For any round n, such that $\phi(n) = (i^*, j)$ for some j, x_n is simulated by the simulator of the protocol. Concretely, For, $i, j \neq i^*$ and $i \neq j$, let $\mathsf{corr}_{i,j} = \mathsf{corr}_{i,j}(F(K_{i,j}, \mathsf{tag})$ and $\mathsf{corr}_{j,i} = \mathsf{corr}_{j,i}(F(K_{i,j}, \mathsf{tag})$. Let $\mathsf{corr}_{j,i^*} = \mathsf{corr}_{j,i^*}(r_j)$ and $\mathsf{corr}_{i^*,j} = \mathsf{corr}_{i^*,j}(r_j)$ for $j \neq i^*$. Now, for any such round n, $x_n$is generated from $Sim(1^\lambda, C(m)\{m_k, \mathsf{corr}_k\}_{k \neq i^*})$.

Informally, the above two hybrids are indistinguishable due to the security of the protocol Γ.
We now argue indistinguishability of the hybrids.

Lemma 1. *If* E *is a secure secret-key encryption scheme, then for any PPT adversary* $\mathcal{A}$*, it holds that* $\Big| Pr\big[\mathcal{A}(\mathsf{Hyb}_1) = 1\big] - Pr\big[\mathcal{A}(\mathsf{Hyb}_2) = 1\big]\Big| < \mathsf{negl}$.

Proof. Let t_C be the number of rounds in the protocol while computing circuit C. The only difference in Hyb_1 and Hyb_2 is the manner in which the ciphertext $\{c_i\}_{i\in[n]}$ are generated. In Hyb_2 we compute $c_n = \mathsf{E.Enc}(sk, x_n, \pi_n)$ where n is any round such that $\phi(n) = (i^*, j)$ or (j, i^*) for some j. Here, sk is the sampled secret-key for the encryption scheme E used while generating function key. For all other rounds n, c_n is generated as an encryption of 0.
In Hyb_1, c_i for all $i \in [t_C]$ are computed as an encryption of 0 (of various lengths). Note that the secret key sk comes up nowhere else in the hybrids and the randomness of the encryptions are never used anywhere in the hybrids. The indistinguishability of the hybrids now follows from the security of the encryption scheme.

Lemma 2. *If* Com *is a hiding commitment scheme, then for any PPT adversary* $\mathcal{A}$*, it holds that* $\Big| Pr\big[\mathcal{A}(\mathsf{Hyb}_2) = 1\big] - Pr\big[\mathcal{A}(\mathsf{Hyb}_3) = 1\big]\Big| < \mathsf{negl}$.

Proof. Let t_C be the number of rounds in the protocol while computing circuit C. The only difference between Hyb_2 and Hyb_3 is the way $Z_{\mathsf{out},i}$ is generated for all $i \in [t_C]$. In Hyb_2 they are generated as commitment of $\perp$ while in Hyb_3, $Z_{\mathsf{out},i}$ such that $\phi(i) = (i^*, j)$ or (j, i^*) for $j \neq i^*$, is generated as a commitment of x_i. The openings of these commitments are not used in these hybrids. The indistinguishability now follows from the security of the commitment scheme. The indistinguishability of the hybrids now follows from the security of the encryption scheme.

Lemma 3. *If for all* $j \neq i^*$*,* $\mathsf{FE}_{i^*,j}$ *and* FE_{j,i^*} *are secure multi-input functional encryption candidates, then for any PPT adversary* $\mathcal{A}$*, it holds that* $\Big| Pr\big[\mathcal{A}(\mathsf{Hyb}_3) = 1\big] - Pr\big[\mathcal{A}(\mathsf{Hyb}_4) = 1\big]\Big| < \mathsf{negl}$.

Proof. Let t_C be the number of rounds in the protocol Γ while computing circuit C. The only difference in Hyb_3 and Hyb_4 is the manner in which the ciphertext $\{\mathsf{CT}_{i,j}\}_{i\neq j}$ are generated. In Hyb_3 they are computed as $\mathsf{CT}_{i,j} = \mathsf{FE}_{i,j}.\mathsf{Enc}(\mathsf{EK}_{i,j,1}, m_i, K'_{i,j}, K_{i,j}, \{Z_{\mathsf{in},k}, Z_{k,i}\}_{k,i\in[n],k\neq i}, r_{i,j}, r_{\mathsf{in},i}, \perp)$

for every $i \neq j$.

In Hyb_4 they are computed as follows: For any $j \neq i^*$, $\mathsf{CT}_{i^*,j} = \mathsf{FE}_{i^*,j}.\mathsf{Enc}$ $(\mathsf{EK}_{i^*,j,1}, m_{i^*}, \perp, \perp, \{Z_{\mathsf{in},k}, Z_{k,i}\}_{k,i\in[n],k\neq i}, \perp, \perp, sk)$ and $\mathsf{CT}_{j,i^*} = \mathsf{FE}_{j,i^*}.$ $\mathsf{Enc}(\mathsf{EK}_{j,i^*,1}, m_j, \perp, \perp, \{Z_{\mathsf{in},k}, Z_{k,i}\}_{k,i\in[n],k\neq i}, \perp, \perp, sk)$. Other cipher-texts are computed as before. Note that since,$\mathsf{NIWI}_{i,j}$ for any $i \neq j$ is statistically sound and the commitment scheme is statistically binding, the function keys issued by $\mathsf{FE}_{i^*,j}$ or FE_{j,i^*} for function G_k such that $\phi(k) = (i^*, j)$ or (j, i^*) these ciphertexts form a valid multi-input functional encryption challenge. The reduction involves $2(n-1)$ intermediate hybrids where we first switch $\mathsf{CT}_{i^*,j}$ one by one for all $j \neq i^*$. Then we switch CT_{j,i^*} one by one for all $j \neq i^*$. The security now follows from the security of $\mathsf{FE}_{i^*,j}$ (or FE_{j,i^*}).

Lemma 4. *If PRF is a secure PRF, then for any PPT adversary $\mathcal{A}$, it holds that* $\Big|Pr\big[\mathcal{A}(\mathsf{Hyb}_4) = 1\big] - Pr\big[\mathcal{A}(\mathsf{Hyb}_5) = 1\big]\Big| < \mathsf{negl}$.

Proof. Let t_C be the number of rounds in the protocol while computing circuit C. The only difference between Hyb_4 and Hyb_5 is the way the hardwired proof π_n is generated for any n such that $\phi(n) = (i^*, j)$ or (j, i^*) for some j. In Hyb_4 they are computed using the randomness $F(K'_{i^*,j}, x_1, .., x_n)$ (or $F(K'_{j,i^*}, x_1, .., x_n)$) while in Hyb_5 it is generated using uniformly sampled randomness. The PRF key does not occur in both the hybrids. The indistinguishability now follows from the security of the PRF scheme.

Lemma 5. *If $\{\mathsf{NIWI}_{j,i^*}, \mathsf{NIWI}_{i^*,j}\}_{j\neq i^*}$ are witness indistinguishable proofs, then for any PPT adversary $\mathcal{A}$, it holds that* $\Big|Pr\big[\mathcal{A}(\mathsf{Hyb}_5) = 1\big] - Pr\big[\mathcal{A}(\mathsf{Hyb}_6) = 1\big]\Big| < \mathsf{negl}$.

Proof. Let t_C be the number of rounds in the protocol while computing circuit C. The only difference between Hyb_5 and Hyb_6 is the way the hardwired proof π_n is generated for any n such that $\phi(n) = (i^*, j)$ or (j, i^*) for some j. Let $\phi(n) = (i', j')$ and either $i' = i^*$ or $j' = i^*$. In Hyb_5, π_n is generated as randomly generated proof using the prover string $\sigma_{i',j'}$ and the genuine witness. In Hyb_6, π_n is generated as randomly generated proof using the prover string $\sigma_{i',j'}$ and the trapdoor witness (i.e. opening of $Z_{\mathsf{out},n}$). The indistinguishability now follows from the witness-indistinguishability property of $\mathsf{NIWI}_{i',j'}$. The reduction introduces intermediate hybrids where π_n ($n \in [t_C]$) is switched one by one to use the trapdoor witness.

Lemma 6. *If Com is a hiding commitment scheme, then for any PPT adversary $\mathcal{A}$, it holds that* $\Big|Pr\big[\mathcal{A}(\mathsf{Hyb}_6) = 1\big] - Pr\big[\mathcal{A}(\mathsf{Hyb}_7) = 1\big]\Big| < \mathsf{negl}$.

Proof. Let t_C be the number of rounds in the protocol while computing circuit C. The only difference between Hyb_6 and Hyb_7 is the way commitments $\{Z_{i^*,j}, Z_{j,i^*}, Z_{\mathsf{in},i^*}\}_{j\neq i^*}$ are generated. In Hyb_6 they are generated as follows: $Z_{i^*,j}$ and Z_{j,i^*} are generated as commitment of K_{j,i^*} for any

$j \neq i^*$. Z_{in,i^*} is computed as a commitment of m_{i^*}.

In Hyb_7 is generated as a commitment of $\perp$. Note that the openings of these commitments are never used in the hybrids. The indistinguishability of the hybrids now follows from the security of the commitment scheme.

Lemma 7. *If PRF is a secure PRF, then for any PPT adversary $\mathcal{A}$, it holds that* $\Big|Pr\big[\mathcal{A}(\mathsf{Hyb}_7) = 1\big] - Pr\big[\mathcal{A}(\mathsf{Hyb}_8) = 1\big]\Big| < \mathsf{negl}$.

Proof. Let t_C be the number of rounds in the protocol while computing circuit C. The only difference between Hyb_7 and Hyb_8 is the way the hardwired input x_i is generated for any round i such that $\phi(i) = (i^*, j)$ or (j, i^*) for some j. In Hyb_7 they are computed using the $\mathsf{corr}_{i^*,j}$(or corr_{j,i^*}) derived from randomness $F(K_{i^*,j}, \mathsf{tag})$ while in Hyb_8 they are computed using uniformly sampled randomness. The PRF keys do not occur anywhere else in the hybrids. The indistinguishability now follows from the security of the PRF scheme.

Lemma 8. *If Γ is secure protocol, then for any PPT adversary $\mathcal{A}$, it holds that* $\Big|Pr\big[\mathcal{A}(\mathsf{Hyb}_8) = 1\big] - Pr\big[\mathcal{A}(\mathsf{Hyb}_9) = 1\big]\Big| < \mathsf{negl}$.

Proof. Let t_C be the number of rounds in the protocol while computing circuit C. The only difference between Hyb_8 and Hyb_9 is the way the hardwired input x_i is generated for any round i such that $\phi(i) = (i^*, j)$ or (j, i^*) for some j. In Hyb_8 they are computed as in the real world experiment while in Hyb_9 they are computed using the simulator as in the ideal experiment. Note that, corr_{i^*} and m_{i^*} does not appear anywhere else in the hybrids. The indistinguishability now follows from the security of the MPC protocol.

Lemma 9. *In Hyb_9 the advantage of the adversary is 0.*

Proof. Since in Hyb_9 m_{i^*} does not appear, due to the information theoretic security of the secret sharing scheme, $\{m_i, \mathsf{corr}_i\}_{i \neq i^*}$ carry no infomation about b. The lemma then follows.

C Some Preliminaries

C.1 Commitment Schemes

We consider commitment schemes secure in the CRS model. We give the definition of this primitive below.

Definition 11 (Commitment scheme in the CRS model). *A polynomial-time computable function* $\mathsf{Com}\colon \{0,1\}^n \times \{0,1\}^\lambda \times \{0,1\}^m \to \{0,1\}^*$, *where $n = \mathsf{poly}(\lambda)$ is the length of the string to commit, λ is the length of the randomness, $m = \mathsf{poly}(\lambda)$ is the length of the CRS. We say that Com is a (non-interactive perfectly binding) commitment scheme in the CRS model if for any two inputs $x_1, x_2 \in \{0,1\}^n$ such that $x_1 \neq x_2$ it holds that:*

1. **Computational Hiding:** *Let* $\mathsf{crs} \leftarrow \{0,1\}^m$ *be chosen uniformly at random. There exists a negligible function* μ *such that random variables* $\mathsf{Com}(x_1, \mathcal{U}_\lambda, \mathsf{crs}) \approx_{c,\mu} \mathsf{Com}(x_2, \mathcal{U}_\lambda, \mathsf{crs})$ *(given* crs*). We say the the scheme is sub-exponentially secure if* $\mu = O(2^{-\lambda^c})$ *for some constant* $c > 0$.
2. **Perfect Binding:** *With all but negligible probability over the CRS, the supports of the above random variables are disjoint.*

Commitment schemes that satisfy the above definition, in the CRS model, can be constructed based on any pseudorandom generator [44] (which can be based on any one-way functions [34]). For simplicity, throghout the paper we ignore the CRS and simply write $\mathsf{Com}(\cdot,\cdot)$. We say that $\mathsf{Com}(x, r)$ is the commitment of the value x with the opening r.
Remark. In the constructions, we omit mentioning CRS of the commitment scheme where it is implied that CRS is generated by the setup algorithm of the primitive. We also note that sub-exponentially secure commitments can be built from sub-exponentially secure one-way functions.

C.2 NIWI Proofs

We will be extensively using one message witness indistinguishable proofs NIWI in the CRS model as provided by [33].

Definition 12. *A pair of PPT algorithms* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *is a* NIWI *for an NP relation* $\mathcal{R}_\mathcal{L}$ *if it satisfies:*
1. *Completeness: for* $\sigma \leftarrow \sigma(1^\lambda)$ *every* $(x, w) \in \mathcal{R}_\mathcal{L}$, $Pr[\mathcal{V}(\sigma, x, \pi) = 1 : \pi \leftarrow \mathsf{Prove}(\sigma, x, w)] = 1$. *The probability is taken over the coins of algorithms* Setup *and* Prove.
2. *(Statistical) Soundness: Proof system is said to be statistically sound if there for every* $x \notin L$ *and* $\pi \in \{0,1\}^*$
$Pr[\mathcal{V}(\sigma, x, \pi) = 1] < 2^{-\lambda}$. *The probability is taken over the coins of* Setup.
3. *Witness indistinguishability: there exists a negligible function* μ *such that for any sequence* $\mathcal{I} = \{(x, w_1, w_2) : w_1, w_2 \in \mathcal{R}_\mathcal{L}(x)\}$
$\{\sigma, \pi_1 : \sigma \leftarrow \mathsf{Setup}(1^\lambda), \pi_1 \leftarrow \mathcal{P}(\sigma, x, w_1)\}_{(x,w_1,w_2)\in\mathcal{I}} \approx_{c,\mu} \{\sigma, \pi_2 : \sigma \leftarrow \mathsf{Setup}(1^\lambda), \pi_2 \leftarrow \mathcal{P}(\sigma, x, w_2)\}_{(x,w_1,w_2)\in\mathcal{I}}$
We say that the system if sub-exponentially secure if $\mu = O(2^{-\lambda^c})$

[33] provides perfectly sound one message witness indistinguishable proofs based on the decisional linear (DLIN) assumption. [8] construct NIWI from one-way functions and indistinguishability obfuscation. Sub-exponentially secure NIWI*s* can be built by assuming sub-exponential security of underlying primitives. We rely on this work for our context.