From FE Combiners to Secure MPC and Back

Prabhanjan Ananth* Saikrishna Badrinarayanan† Aayush Jain†
Nathan Manohar† Amit Sahai†

Abstract

Functional encryption (FE) has incredible applications towards computing on encrypted data. However, constructing the most general form of this primitive has remained elusive. Although some candidate constructions exist, they rely on nonstandard assumptions, and thus, their security has been questioned. An FE combiner attempts to make use of these candidates while minimizing the trust placed on any individual FE candidate. Informally, an FE combiner takes in a set of FE candidates and outputs a secure FE scheme if at least one of the candidates is secure.

Another fundamental area in cryptography is secure multi-party computation (MPC), which has been extensively studied for several decades. In this work, we initiate a formal study of the relationship between functional encryption (FE) combiners and secure multi-party computation (MPC). In particular, we show implications in both directions between these primitives. As a consequence of these implications, we obtain the following main results.

- A two round semi-honest MPC protocol in the plain model secure against up to $n-1$ corruptions with communication complexity proportional only to the depth of the circuit being computed assuming LWE. Prior two round protocols that achieved this communication complexity required a common reference string.

- A functional encryption combiner based on pseudorandom generators (PRGs) in NC^1. Such PRGs can be instantiated from assumptions such as DDH and LWE. Previous constructions of FE combiners were known only from the learning with errors assumption. Using this result, we build a universal construction of functional encryption: an explicit construction of functional encryption based only on the assumptions that functional encryption exists and PRGs in NC^1.

1 Introduction

Functional encryption (FE), introduced by [SW05, BSW11, O'N10], is one of the core primitives in the area of computing on encrypted data. This notion allows the authority to generate and

*MIT. prabhanjan@csail.mit.edu.

†UCLA and Center for Encrypted Functionalities. {saikrishna, aayushjain, nmanohar, sahai}@cs.ucla.edu.
Research supported in part by a DARPA/ARL SAFEWARE award, NSF Frontier Award 1413955, NSF grants 1619348, 1228984, 1136174, and 1065276, BSF grant 2012378, 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 authors and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

distribute constrained keys associated with functions $f_1, \ldots, f_q$, called *functional keys*, which can be used to learn the values $f_1(x), \ldots, f_q(x)$ given an encryption of x. Intuitively, the security notion states that the functional keys associated with $f_1, \ldots, f_q$ and an encryption of x reveal nothing beyond the values $f_1(x), \ldots, f_q(x)$. While this notion is interesting on its own, several works have studied its connections to other areas in cryptography and beyond, including reusable garbled circuits [GKP+13], indistinguishability obfuscation [AJ15, BV15, AJS15, LPST16b, LPST16a], adaptive garbling [HJO+16], verifiable random functions [GHKW17, Bit17, BGJS17], deniable encryption [GKW17], hardness of Nash equilibrium [GPS16, GPSZ17] and many more.

Currently, we know how to construct only restricted versions[1] of functional encryption from well studied cryptographic assumptions. However, constructing the most general form of functional encryption has been an active research area and has intensified over the past few years given its implication to indistinguishability obfuscation [AJ15, BV15]. All the candidates [GGHZ14, Lin16, LV16, AS17, Lin17, LT17] we know so far are based on assumptions pertaining to the tool of graded encodings [BS03, GGH13]. Recent cryptanalytic attacks [CHL+15, HJ15, CLLT15, CJL16, CGH+15] on assumptions related to graded encodings have prompted scrutiny of the security of schemes that use this tool as the building block. Given this, we should hope to minimize the trust we place on any individual FE candidate. The notion of a functional encryption combiner achieves this purpose. Roughly speaking, a functional encryption combiner allows for combining many functional encryption candidates in such a way that the resulting FE candidate is secure as long as any one of the initial FE candidates is secure. In other words, a functional encryption combiner says that it suffices to place trust collectively on multiple FE candidates, instead of placing trust on any specific FE candidate.

We initiate a systematic study of functional encryption combiners. In particular, we study a formal relation between functional encryption combiners and *secure multi-party computation*, a fundamental notion in cryptography. Secure multi-party computation [Yao86, GMW87, BOGW88] allows for many parties, who don't necessarily trust each other, to come together and compute a function on their private inputs. By studying implications from FE combiners to secure MPC (and vice versa), we achieve interesting consequences that were previously unknown. We believe that our results will lead to the ability to translate any advances in one area into corresponding advances in the other area. We detail our contributions next.

1.1 Our Contributions

Our results can be classified into two parts. The first part shows how to translate constructions of functional encryption combiners into secure MPC protocols. The second part studies the other direction.

From FE combiners to secure MPC: Our first result shows how to construct a passively secure multi-party computation protocol that is both round optimal (two rounds) and communication efficient (depends only on circuit depth). Moreover, our resulting protocol is in the plain model and can tolerate all but one corruption[2]. Prior round optimal passively secure MPC protocols were either communication inefficient, that is communication complexity was proportional to circuit size [GS17, BGI+18, BL18, GS18], or were based on trust assumptions [CM15, MW16, PS16, BP16] (for instance, a common reference string). We prove the following theorem.

[1] For instance, we can restrict the adversary to only ask for one functional key in the security experiment. A functional encryption scheme satisfying this property can be based on public key encryption schemes [SS10, GVW12] (or one-way functions if one can settle for the secret key version).

[2] Unless otherwise specified, we only consider MPC protocols tolerating all but one corruption.

Theorem 1 (Informal). *Consider an n-party functionality f, for any $n \geq 2$. Assuming learning with errors, there is a construction of a passively secure (semi-honest) n-party computation protocol for f in the plain model secure against up to $n-1$ corruptions. The number of rounds in this protocol is 2, and the communication complexity is $\mathsf{poly}(\lambda, d, L)$, where d is the depth of the circuit computing f and L is the total length of the inputs of all the parties.*

We note that such communication-efficient MPC protocols imply fully homomorphic encryption and thus, a construction from an assumption other than learning with errors would translate to a construction of fully homomorphic encryption from the same assumption.

Central to proving the above theorem is a transformation from a functional encryption combiner to passively secure MPC. We assume some structural properties on the functional encryption combiner. The functional key for f associated with the combined candidate needs to be of the form $(f, sk_f^1, \ldots, sk_f^n)$, where (i) decomposability: sk_f^i is produced by the i^{th} FE candidate and, (ii) succinctness: the length of sk_f^i is $\mathsf{poly}(\lambda, d)$, where d is the depth of the circuit computing f. As part of the succinctness property, we also require that the encryption complexity is $\mathsf{poly}(\lambda, d)$.

An intermediate tool we use in this implication is a communication **in**efficient passively secure MPC protocol. By communication inefficient, we mean that the communication complexity is proportional to the size of the circuit representing f. We note that such protocols [GS17, BL18, GS18] exist in the literature[3] based on just the assumption of round optimal passively secure oblivious transfer.

Lemma 1 (Informal). *Consider a n-party functionality f, for any $n \geq 2$. There is a passively secure n-party computation protocol for f in two rounds with communication complexity $\mathsf{poly}(\lambda, d, L)$, where d and L are as defined above. Moreover, we assume (i) a decomposable and succinct functional encryption combiner and (ii) a communication inefficient (as defined above) two-round secure n-party computation protocol.*

We show how to construct a functional encryption combiner satisfying the above two properties assuming learning with errors. By plugging the recent round optimal secure MPC protocols [GS17, BL18, GS18] that can be based on two round oblivious transfer, which in turn can be based on learning with errors [PVW08], we get Theorem 1.

From secure MPC to FE combiners: In the other direction, we show how to transform existing secure multi-party computation protocols into constructions of functional encryption combiners. In particular, we show how to transform specific constant round passively secure MPC protocols based on low degree randomized encodings [BMR90] into functional encryption combiners. By instantiating low degree randomized encodings from pseudorandom generators in NC^1, we get the following result.

Theorem 2 (Informal). *Assuming pseudorandom generators in NC^1, there is a construction of an unbounded functional encryption combiner.*

We note that such pseudorandom generators in NC^1 can be instantiated with assumptions such as DDH [NR04] and learning with errors [BPR12]. Next, we present a generic reduction that can transform two round passively secure MPC protocols into functional encryption combiners. For this transformation to hold, the MPC protocol must satisfy two properties: (i) delayed function-dependence: the first round of the MPC protocol should be independent of the functionality

[3]These protocols are inherently communication inefficient. The reason is that they present a compiler that turns any arbitrary interactive MPC protocol into a two round MPC protocol. The communication complexity in the resulting two round MPC protocol is at least the computational complexity of the original MPC protocol. However, the computational complexity of the resulting protocol has to be proportional to the size of the circuit representing the functionality f.

being securely computed and (ii) reusability: the first round can be reused by the parties to securely compute many functionalities (but on the same inputs fixed by the first round).

Theorem 3 (Informal)**.** *Assuming a delayed function-dependent and reusable round optimal secure MPC protocol, there is a construction of an unbounded decomposable functional encryption combiner.*

We then observe that existing two round secure MPC protocols [MW16, PS16, BP16], based on learning with errors, already satisfy delayed function-dependence and reusability. We note that it is not necessary for the round optimal protocols to be in the plain model (indeed, the protocols [MW16, PS16, BP16] are in the common reference string (CRS) model).

Prior to this work, the only polynomial hardness assumption known to imply an FE combiner was the learning with errors assumption [AJS17]. While Theorem 2 already gives a construction of a functional encryption combiner from learning with errors (pseudorandom generators in NC^1 can be based on learning with errors [BPR12]), the functional encryption combiner constructed in Theorem 3 arguably provides a more efficient transformation. In particular, the efficiency of the functional keys in the combined scheme from Theorem 3 is linear in the efficiency of the functional keys in the FE candidates. However, the efficiency in the combined scheme from Theorem 2 degrades polynomially in the efficiency of the original FE candidates. Furthermore, the FE combiner from Theorem 3 is *decomposable*, a property needed by an FE combiner as a building block in the proof of Theorem 1. On the other hand, the FE combiner from Theorem 2 is inherently not decomposable, since it is based on an "onion-layered" approach. Furthermore, the FE combiner from Theorem 3 makes only black-box use of the underlying FE candidates, whereas the FE combiner from Theorem 2 is inherently non-black-box.

In terms of techniques, we introduce mechanisms to emulate a MPC protocol using functional encryption candidates. This is reminiscent of "MPC-in-the-head" paradigm introduced by Ishai et al. [IKOS07] and more relevant to the context of FE is the work of Gorbunov et al. [GVW12] who used information-theoretic MPC protcols to construct single-key FE. However, we encounter new challenges to implement the "MPC-in-the-head" paradigm in our context.

Universal Functional Encryption: We strengthen our constructions of FE combiners by showing how to transform them into combiners that also work when the insecure candidates don't necessarily satisfy correctness (of course, we still require that the secure candidate is correct). Such combiners are called *robust combiners*. To do this, we present correctness amplification theorems based on previous works on indistinguishability obfuscation [BV16, AJS17] and, in particular, our correctness amplification assumes only one-way functions (unlike [BV16, BV17]). Robust combiners have been useful in universal constructions [AJN+16, AJS17]. Roughly speaking, a universal construction of FE is a concrete construction of FE that is secure as long as any secure and correct construction exists. We show how to build universal functional encryption from robust FE combiners.

Theorem 4 (Universal Functional Encryption)**.** *Assuming pseudorandom generators in* NC^1*, there is a universal unbounded functional encryption scheme.*

Our construction will be parameterized by T, where T is an upper bound on the running time of all the algorithms associated with the secure candidate. This was a feature even in the universal iO construction of [AJN+16].

Related Work: The notion of combiners has been studied in the context of many cryptographic primitives. Asmuth and Blakely [AB81] studied combiners for encryption schemes. Levin proposed a universal construction of one-way functions [Lev87]. Later, a systematic study of combiners and their relation to universal constructions was proposed by Harnik et

al. [HKN+05] (also relevant are the constructions in [Her05, Her09]). Recently, Ananth et al. [AJN+16] designed universal constructions of indistinguishability obfuscation (iO). Concurrently, Fishclin et al. also proposed combiners in the context of program obfuscation [FHNS16]. Subsequently, Ananth et al. [AJS17] proposed the concept of transforming combiners that transforms many candidates of a primitive X, with at least one of them being secure and, into a secure candidate of primitive Y. In particular, they construct iO-to-functional encryption transforming combiners.

1.2 Technical Overview

We begin by tackling the problem of constructing secure multi-party computation with depth-proportional communication complexity, i.e, proportional only to the depth of the circuit being securely computed, starting from a functional encryption combiner.

Round Optimal MPC with Depth-Proportional Communication: Let's start by recalling prior known two round secure MPC protocols [MW16, PS16, BP16] with depth-proportional communication in the CRS model. The basic template is as follows: in the first round, the i^{th} party broadcasts an encryption of its input x_i. These ciphertexts are computed with respect to public keys that are derived from the CRS. All the n parties then homomorphically compute on the encryptions of $(x_1, \ldots, x_n)$ to obtain a ciphertext of $f(x_1, \ldots, x_n)$, where f is the function they wish to securely compute. The resulting ciphertext is then partially decrypted, and every party broadcasts its partially decrypted value in the second round. These values can be combined to recover the output of the functionality.

One could imagine getting rid of the CRS in the above protocol using the recent round optimal MPC protocols in the plain model [GS17, BL18]. If this were possible, then it would yield a round optimal MPC in the plain model that has depth-proportional communication complexity. However, the issue is that the messages in the first round of [MW16, PS16, BP16] are computed as functions of the CRS and thus, such an approach would inherently require three rounds.

To overcome this, we introduce a mechanism to parallelize the evaluation and the encryption processes. The output of the evaluation in our approach is the output of the functionality and not a partially decrypted value, as was the case in [MW16, PS16, BP16], and thus, we save one round. To implement this high level idea, we resort to a functional encryption combiner. Before we describe the high level template, we require that the underlying functional encryption combiner satisfies the *decomposability property*: Suppose we have FE candidates $\mathsf{FE}_1, \ldots, \mathsf{FE}_n$. Then, a functional key for a circuit C in the combined scheme is just a concatenation of the functional keys for C, $(sk_1^C, \ldots, sk_n^C)$, where sk_i^C is computed with respect to the i^{th} FE candidate.

The template of our depth-proportional communication MPC construction from an FE combiner satisfying this decomposability property is as follows:

- Suppose the input of the i^{th} party is x_i, and f is the function to be securely computed. All the parties execute the two round (communication-inefficient) MPC protocol from [GS17, BL18] to obtain an encryption of $(x_1, \ldots, x_n)$ with respect to the combined FE scheme.
- Simultaneously, the i^{th} party computes functional key of f with respect to the i^{th} candidate and sends it to everyone.

At the end of second round, every party has an encryption of $(x_1, \ldots, x_n)$ with respect to the combined candidate and functional keys for f with respect to every candidate. From the decomposability property, this is equivalent to generating functional key for f with respect to the combined candidate. Each party can separately execute the FE decryption algorithm to

obtain $f(x_1, \ldots, x_n)$, as desired. Here, we crucially rely on the fact that all the FE candidates are correct.

In terms of security, we could hope to argue that as long as one of the FE candidates is secure, then the resulting MPC protocol is also secure. However, this is not true. It could be the case that only the candidate FE_i is secure while the adversary corrupts all the parties except the j^{th} party, with $i \neq j$. In this case, the security of FE_i does not reduce to the security of MPC. The idea here is to start with the assumption of a secure FE scheme and instantiate all the candidates using the same FE scheme. This means that either all the FE candidates are secure or none of them are. If the adversary corrupts all but the j^{th} party, this means that he can obtain all the master secret keys of the FE scheme except the j^{th} one. This is effectively the same as all except the j^{th} candidate being broken. At this point, we can use the security of the j^{th} FE scheme to argue the security of the MPC protocol. This shows that the above template yields a secure two round MPC protocol assuming a secure FE scheme.

Note that we also assume a two round (communication-inefficient) MPC protocol. Without showing that our protocol has depth-proportional communication, the above protocol doesn't achieve anything. Indeed, it is unclear why our protocol should have depth-proportional communication. There are two sources of concern: (i) we are still using a communication *in*efficient MPC protocol and, (ii) the functional key of f could be proportional to the size of the circuit computing f. Suppose we had a secure (magical) FE scheme satisfying the following two properties: (1) the encryption complexity of this FE scheme is proportional only to the depth of f and, (2) the functional key of f is of the form (f, aux), where $|\mathsf{aux}|$ only depends on the depth of the circuit computing f. We claim that this would immediately show that our protocol has communication complexity proportional only to the depth. Concern (i) is addressed by the fact the communication-inefficient MPC protocol is used only to evaluate the encryption circuit of the underlying FE scheme. Since the underlying FE scheme is succinct, the size of the encryption circuit only depends on the depth of the functionality f. Therefore, the communication complexity of the communication-inefficient MPC protocol does not affect our construction. Concern (ii) is handled by the fact that the parties only have to send the "aux" part of the function keys to the other parties, which is only proportional to the depth of f.

It is unclear why such a (magical) FE scheme should exist. We observe that the functional encryption scheme of Goldwasser et al. [GKP+13] satisfies both properties (1) and (2). We recall the functional encryption construction of Goldwasser et al.: the building blocks in this construction are attribute based encryption (ABE) for circuits, fully homomorphic encryption (FHE), and garbling schemes.

- To encrypt a message x, first encrypt x using a (leveled) FHE scheme. Suppose the maximum output length of the functions for which we generate functional keys is L. Generate $\mathsf{poly}(L)$ ABE encryptions of the FHE ciphertext, for some fixed polynomial poly, along with wire keys of a garbled circuit. The garbled circuit is associated with the FHE decryption circuit.
- The functional keys of f consists of many ABE keys associated with the circuit that computes the FHE evaluation of f.

If we instantiate the ABE scheme with the scheme of Boneh et al. [BGG+14] and the leveled FHE scheme with any of the schemes proposed in [GSW13, BGV14], we achieve both properties (1) and (2) described above. The schemes of [BGG+14] and [GSW13, BGV14] have encryption complexity proportional only to the depth of the circuit. In terms of the structure of the functional key, we note that the ABE scheme of [BGG+14] satisfies this nice property: you can express the ABE key of a function f as (f, aux), where $|\mathsf{aux}|$ is a polynomial in depth. This can be used to argue that the above FE scheme satisfies property (2).

Thus starting from an FE combiner, we have constructed a communication-efficient two round MPC. We note that the FE combiner is required to satisfy simulation security in order

to prove that the resulting MPC is simulation secure. The security proof of the resulting MPC directly follows from the simulation security of the FE combiner and the simulation security of the underlying communication inefficient MPC.

Next, we show how to construct such an FE combiner.

MPC to FE combiner: As in the works of [AJN+16, AJS17], we view the FE candidates as analogous to parties in a secure MPC protocol. Suppose we want to construct an FE combiner for n candidates. We start with a two round (semi-honest) secure n-party MPC protocol in the plain model. To encrypt a message x, first additively secret share x into shares $(x_1, \ldots, x_n)$. Compute the first round messages of all the parties, where the i^{th} party's input is x_i. Finally, for every $i \in [n]$, encrypt the first round messages of all the parties along with the local state of the i^{th} party using i^{th} FE candidate. All the n encryptions will form the ciphertext corresponding to the FE combiner scheme.

To generate a functional key for f, we generate n functional keys with each key associated with an FE candidate. The i^{th} functional key computes the next message function of the i^{th} party. In this context, we define the next message function to be a deterministic algorithm that takes as input the state of the party along with the messages received so far and produces the next message. Moreover, the MPC functionality associated with the next message function is as follows: it takes as input n shares of x, reconstructs x, and computes $f(x)$. The functional key of f corresponding to the FE combiner is the collection of all these n functional keys.

The decryption in the FE combiner scheme proceeds by recovering the first and second round messages of all the parties. The reconstruction algorithm of the secure MPC protocol is then executed to recover the output of the functionality. An issue here is that the reconstruction part need not be publicly computable. Meaning that it might not be possible to recover the output of the functionality from the transcript of the protocol alone. This can be resolved by revealing the local state of one of the parties to the FE evaluator who can then use this to recover the output. We implement this by considering an $(n+1)$-party MPC protocol with the FE evaluator corresponding to one of the parties in the MPC protocol.

Without restricting ourselves to a specific type of two round secure MPC protocols, the above template could be ill defined for two reasons:

- *Function-Dependence*: The first round messages of the MPC protocol we start off with could depend on the functionality being securely computed. This means that the FE encryptor needs to be aware of the function f when it is encrypting the message x. Hence, we need to enforce a delayed function-dependence property on the underlying MPC protocol. Roughly, this property states that the first round messages of the MPC protocol are independent of the functionality being securely computed.
- *Reusability*: Suppose we wish to construct a *collusion-resistant* FE combiner, meaning that the FE combiner is secure even if the adversary obtains multiple functional keys during the security experiment. Even if one of the candidates is secure in the collusion-resistant setting, the above template doesn't necessarily yield a collusion-resistant FE combiner. This is because the first round MPC messages are "reused" across different FE evaluations. The security of MPC, as is, doesn't necessarily guarantee any security if the first round messages are reused for secure computation of multiple functionalities. Hence, we need to enforce a corresponding reusability property on the underlying MPC protocol to make it work in the collusion resistant setting.

Once we start with a delayed function-dependent and reusable secure MPC protocol, we can implement a FE combiner using the above template. We observe that the schemes of [MW16, PS16, BP16] are both delayed function-dependent and reusable. As a corollary, we obtain an FE combiner based on learning with errors.

We note that this would give an FE combiner that satisfies indistinguishability security. This is inherent since collusion-resistant FE that is also simulation secure was shown to be impossible [AGVW13]. Thus, for our application of communication efficient MPC, we construct a simulation secure FE combiner in the single-key setting (i.e., the adversary can only submit one function query) starting from a threshold fully homomorphic encryption scheme.

FE Combiner from Weaker Assumptions: The above constructions and previous constructions of FE combiners [AJS17] relied on the learning with errors assumption. However, it would be interesting to try to construct an FE combiner from weaker assumptions. Our first observation is that there is a simple construction of an FE combiner for two FE candidates. In this case, one can simply "nest" the two candidates. That is, if the candidates are denoted FE_1 and FE_2, we encrypt a message x by first encrypting x under FE_1 and then encrypting the resulting ciphertext under FE_2. To generate a function key for f, we first generate the function key SK_1 for f using FE_1 and then generate the function key SK_2 for the decryption circuit of FE_1, with SK_1 hardcoded as the function key, using FE_2. SK_2 is then the function key for f in the nested scheme. In fact, this nested approach works to combine any constant d number of candidates. However, this approach does not scale polynomially in the number of candidates, and therefore, does not give us an FE combiner for a polynomial number of candidates.

Using the above observation, we note that we can evaluate circuits over a constant number of inputs. In particular, we can evaluate constant-sized products. If we could compute the sum of various constant-sized products, then we could compute constant-degree polynomials, which would allow us to apply known bootstrapping techniques to go from FE for constant degree polynomials to FE for arbitrary functions via randomized encodings. Such randomized encodings can be constructed assuming a PRG in NC^1 [AIK05]. But how do we go about computing the sums of constant degree polynomials? To reason about this, we will view this as an MPC problem, where each FE candidate is associated with a party. Given an input x, we bitwise secret share x amongst all the parties. This effectively gives us an MPC problem where each party/candidate has a secret input (their share of x). For simplicity, let's consider the case where each candidate is given a single bit (the ith candidate is given the bit x_i). As an example, suppose we wished to evaluate the polynomial

$$x_1^2 + x_1x_2 + x_1x_3 + x_2x_3.$$

Using the simple nested combiner for two candidates, we could evaluate each monomial and then sum the resulting monomial evaluations to compute the polynomial. However, this approach is flawed, since it will leak the values of each of the monomials, whereas functional encryption requires *only* the value of the polynomial to be computable and nothing else. We resolve this issue by masking each of the monomial evaluations by secret shares of 0 such that summing all these values gives the correct polynomial evaluation, but the individual computed monomial evaluations hide the true values of the monomials. To illustrate this, for the above polynomial, candidate 1 has its secret input in 3 monomials x_1^2, x_1x_2, and x_1x_3. We secret share 0 across 3 shares. Let $\mathsf{Share}_{1,1}, \mathsf{Share}_{1,2}, \mathsf{Share}_{1,3}$ denote these values, where

$$\mathsf{Share}_{1,1} + \mathsf{Share}_{1,2} + \mathsf{Share}_{1,3} = 0.$$

Similarly, candidates 2 and 3 have their secret inputs in 2 monomials: x_1x_2, x_2x_3 for candidate 2 and x_1x_3, x_2x_3 for candidate 3. We secret share 0 across 2 shares for each of these candidates. These shares are denoted $\mathsf{Share}_{2,1}, \mathsf{Share}_{2,2}$ for candidate 2 and $\mathsf{Share}_{3,1}, \mathsf{Share}_{3,2}$ for candidate 3. We then place a total ordering on the monomials of the polynomial in order to assign the shares to the monomials. Suppose our ordering was

$$x_1^2 < x_1x_2 < x_1x_3 < x_2x_3.$$

Then, we would see that x_1^2 was the first monomial containing x_1 and assign $\mathsf{Share}_{1,1}$ to this monomial. For x_1x_2, we see that it is the second monomial containing x_1 and the first monomial containing x_2. Therefore, we assign the shares $\mathsf{Share}_{1,2}$ and $\mathsf{Share}_{2,1}$ to the monomial x_1x_2. In a similar manner, we assign the shares $\mathsf{Share}_{1,3}, \mathsf{Share}_{3,1}$ to x_1x_3 and the shares $\mathsf{Share}_{2,2}, \mathsf{Share}_{3,2}$ to x_2x_3. When generating the function key to evaluate the monomial x_1^2, we actually give out a function key that evaluates $x_1^2 + \mathsf{Share}_{1,1}$. Similarly, when generating a function key to evaluate the monomial x_1x_2, we actually give out a function key that evaluates $x_1x_2 + \mathsf{Share}_{1,2} + \mathsf{Share}_{2,1}$.

By proceeding in this manner, we have made it so that each monomial evaluation hides the actual monomial value, but the sum of the monomial evaluations gives the polynomial value. However, this approach still raises several concerns: (i) our secret sharing procedure doesn't necessarily hide intermediate sums of monomials, and (ii) it is unclear how to coordinate the randomness needed to generate the secret shares amongst the various monomials. To illustrate the first issue, suppose that the polynomial to evaluate was $x_1 + x_2$. In this instance, we would not add any secret shares, which would reveal x_1 and x_2. Fortunately, the first issue is not an issue at all, since such problematic polynomials will not occur. This is because we begin by secret sharing the bits of the input x amongst the candidates. Therefore, every monomial will be broken into the sum of new monomials, such that each candidate contains a private bit in one of these new monomials. Since one of the candidates is secure, the secret sharing amongst the monomials with bits corresponding to the secure candidate ensures that nothing except the actual polynomial evaluation can be learned. To solve issue (ii), we utilize a PRF and generate a random PRF key for each candidate. This PRF key is then used to generate the secret shares of 0 associated with that candidate.

Organization: We begin by defining the notion of functional encryption and secure multi-party computation in Section 2. In Section 3, we define the notion of a functional encryption combiner. In Section 4, we show how to build a decomposable FE combiner that will be used as a building block in the construction of our round optimal and communication efficient MPC protocol and how to instantiate it from [GKP+13]. In Section 5, we give the construction of our round optimal and communication efficient MPC protocol. In Section 6, we show how to build an FE combiner assuming the existence of a PRG in NC^1. In Section 7, we demonstrate how to convert a delayed function-dependent and reusable round optimal secure MPC protocol into an FE combiner. Finally, in Section 8, we show how to convert an FE combiner into a robust FE combiner and build a universal functional encryption scheme.

2 Preliminaries

We denote the security parameter by λ. For an integer $n \in \mathbb{N}$, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. We use $\mathcal{D}_0 \cong_c \mathcal{D}_1$ to denote that two distributions $\mathcal{D}_0, \mathcal{D}_1$ are computationally indistinguishable. We use $\mathsf{negl}(\lambda)$ to denote a function that is negligible in λ. We use $x \leftarrow \mathcal{A}$ to denote that x is the output of a randomized algorithm $\mathcal{A}$, where the randomness of $\mathcal{A}$ is sampled from the uniform distribution.

2.1 Functional Encryption

We define the notion of a (secret key) functional encryption candidate and a (secret key) functional encryption scheme. A functional encryption candidate is associated with the correctness requirement, while a secure functional encryption scheme is associated with both correctness and security.

Syntax of a Functional Encryption Candidate/Scheme. A functional encryption (FE) candidate/scheme FE for 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 as follows. Let $\mathcal{X}_\lambda$ be the input space of the circuit class $\mathcal{C}_\lambda$ and let $\mathcal{Y}_\lambda$ be the output space of $\mathcal{C}_\lambda$. We refer to $\mathcal{X}_\lambda$ and $\mathcal{Y}_\lambda$ as the input and output space of the candidate/scheme, respectively.

- **Setup,** $\mathsf{MSK} \leftarrow \mathsf{FE.Setup}(1^\lambda)$: It takes as input the security parameter λ and outputs the master secret key MSK.
- **Encryption,** $\mathsf{CT} \leftarrow \mathsf{FE.Enc}(\mathsf{MSK}, m)$: It takes as input the master secret key MSK and a message $m \in \mathcal{X}_\lambda$ and outputs CT, an encryption of m.
- **Key Generation,** $\mathsf{SK}_C \leftarrow \mathsf{FE.KeyGen}(\mathsf{MSK}, C)$: It takes as input the master secret key MSK and a circuit $C \in \mathcal{C}_\lambda$ and outputs a function key SK_C.
- **Decryption,** $y \leftarrow \mathsf{FE.Dec}(\mathsf{SK}_C, \mathsf{CT})$: It takes as input a function secret key SK_C, 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, $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ can be represented as Turing machines (or equivalently uniform circuits).

We describe the properties associated with the above candidate.

Approximate Correctness.

Definition 1 (Approximate Correctness)**.** *A functional encryption candidate* $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is said to be* α*-correct if it satisfies the following property: for every* $C : \mathcal{X}_\lambda \to \mathcal{Y}_\lambda \in \mathcal{C}_\lambda, m \in \mathcal{X}_\lambda$ *it holds that:*

$$\Pr\left[\begin{array}{c} \mathsf{MSK} \leftarrow \mathsf{FE.Setup}(1^\lambda) \\ \mathsf{CT} \leftarrow \mathsf{FE.Enc}(\mathsf{MSK}, m) \\ \mathsf{SK}_C \leftarrow \mathsf{FE.KeyGen}(\mathsf{MSK}, C) \\ C(m) \leftarrow \mathsf{FE.Dec}(\mathsf{SK}_C, \mathsf{CT}) \end{array}\right] \geq \alpha,$$

where the probability is taken over the coins of the algorithms.

We refer to FE candidates that satisfy the above definition of correctness with $\alpha = 1 - \mathsf{negl}(\lambda)$ *for a negligible function* $\mathsf{negl}(\cdot)$ *as (almost) correct candidates.*

Except for Section 8, we will only deal with correct candidates. Unless explicitly stated otherwise, all FE candidates throughout this paper satisfy (almost) correctness.

We refer the reader to Appendix A.1 for the indistinguishability and simulation based security definitions.

Collusions. We can parameterize the FE candidate by the number of function secret key queries that the adversary can make in the security experiment. If the adversary can only submit an a priori upper bounded q secret key queries, we say that the scheme is q-key secure. We say that the functional encryption scheme satisfies security against an unbounded collusion if the adversary can make an unbounded (polynomial) number of function secret key queries. In this work, unless otherwise stated, we will allow the adversary to make an arbitrary polynomial number of function secret key queries.

Succinctness.

Definition 2 (Succinctness)**.** *A functional encryption candidate* $\mathsf{FE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ *for a circuit class* $\mathcal{C}$ *containing circuits that take inputs of length* ℓ_{in}*, outputs strings of length* ℓ_{out} *bits and are of depth at most* d *is said to be succinct if the following holds: For any circuit* $C \in \mathcal{C}$*,*

- *Let* $\mathsf{MSK} \leftarrow \mathsf{FE.Setup}(1^\lambda)$*. The size of the circuit* $\mathsf{FE.Enc}(\mathsf{MSK}, \cdot) < \mathsf{poly}(\lambda, d, \ell_{\mathsf{in}}, \ell_{\mathsf{out}})$ *for some polynomial* poly*.*
- *The function key* $\mathsf{SK}_C \leftarrow \mathsf{FE.KeyGen}(\mathsf{MSK}, C)$ *is of the form* (C, aux) *where* $|\mathsf{aux}| \leq \mathsf{poly}(\lambda, d, \ell_{\mathsf{out}})$ *for some polynomial* poly*.*

In general, an FE candidate/scheme need not satisfy succinctness. However, we will need to utilize succinct FE candidates when constructing depth-proportional communication MPC (Section 4 and Section 5). In such cases, we will explicitly state that the FE candidates are succinct.

FE Candidates vs. FE Schemes. As defined above, an FE scheme must satisfy *both* correctness and security, while an FE candidate is simply the set of algorithms. Unless otherwise specified, we will be dealing with FE candidates that satisfy correctness. We will only refer to FE constructions as FE schemes if it is known that the construction satisfies both correctness and security.

2.2 Secure Multi-Party Computation

The syntax and security definitions for secure multi-party computation can be found in Appendix A. Since we are dealing throughout this paper with the efficiency of MPC protocols, we give the definition of a succinct MPC protocol below.

Definition 3 (Succinct MPC protocol)**.** *Consider an* n*-party semi-honest secure MPC protocol* Π *for a functionality* f*, represented by a polynomial-sized circuit* C*. We define the communication complexity of* Π *to be the total length of all the messages exchanged in the protocol.*

We define Π *to be succinct if the communication complexity of* Π *is* $\mathsf{poly}(\lambda, d, n)$*, where* λ *is the security parameter and* d *is the depth of the circuit* C*.*

2.3 Additional Preliminaries

In this work, we will also make occasional use of threshold leveled fully homomorphic encryption [AJLA+12, MW16, BGG+17] and garbling schemes [Yao86, BHR12]. Formal definitions of these primitives can be found in Appendix A.

3 FE Combiners: Definition

In this section, we give a formal definition of an FE combiner. Intuitively, an FE combiner FEComb takes n FE candidates, $\mathsf{FE}_1, \ldots, \mathsf{FE}_n$ and compiles them into a new FE candidate with the property that FEComb is a secure FE scheme provided that at least one of the n FE candidates is a secure FE scheme.

Syntax of a Functional Encryption Combiner. A functional encryption combiner FEComb for 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 as follows. Let $\mathcal{X}_\lambda$ be the input space of the circuit class $\mathcal{C}_\lambda$ and let $\mathcal{Y}_\lambda$ be the output space of $\mathcal{C}_\lambda$. We refer to $\mathcal{X}_\lambda$ and $\mathcal{Y}_\lambda$ as the input and output space of the combiner, respectively. Furthermore, let $\mathsf{FE}_1, \ldots, \mathsf{FE}_n$ denote the descriptions of n FE candidates.

- **Setup,** $\mathsf{FEComb.Setup}(1^\lambda, \{\mathsf{FE}_i\}_{i\in[n]})$: It takes as input the security parameter λ and the descriptions of n FE candidates $\{\mathsf{FE}_i\}_{i\in[n]}$ and outputs the master secret key MSK.

- **Encryption,** $\mathsf{FEComb.Enc}(\mathsf{MSK}, \{\mathsf{FE}_i\}_{i\in[n]}, m)$: It takes as input the master secret key MSK, the descriptions of n FE candidates $\{\mathsf{FE}_i\}_{i\in[n]}$, and a message $m \in \mathcal{X}_\lambda$ and outputs CT, an encryption of m.

- **Key Generation,** $\mathsf{FEComb.Keygen}\left(\mathsf{MSK}, \{\mathsf{FE}_i\}_{i\in[n]}, C\right)$: It takes as input the master secret key MSK, the descriptions of n FE candidates $\{\mathsf{FE}_i\}_{i\in[n]}$, and a circuit $C \in \mathcal{C}_\lambda$ and outputs a function key SK_C.

- **Decryption,** $\mathsf{FEComb.Dec}\left(\{\mathsf{FE}_i\}_{i\in[n]}, \mathsf{SK}_C, \mathsf{CT}\right)$: It is a deterministic algorithm that takes as input the descriptions of n FE candidates $\{\mathsf{FE}_i\}_{i\in[n]}$, a function secret key SK_C, and a ciphertext CT and outputs a value $y \in \mathcal{Y}_\lambda$.

Remark 1. *In the formal definition above, we have included* $\{\mathsf{FE}_i\}_{i\in[n]}$*, the descriptions of the FE candidates, as input to all the algorithms of* FEComb*. For notational simplicity, we will often forgo these inputs and assume that they are implicit.*

We now define the properties associated with an FE combiner. The three properties are correctness, polynomial slowdown, and security. Correctness is analogous to that of an FE candidate, provided that the n input FE candidates are all valid FE candidates. Polynomial slowdown says that the running times of all the algorithms of FEComb are polynomial in λ and n. Finally, security intuitively says that if at least one of the FE candidates is also secure, then FEComb is a secure FE scheme. We provide the formal definitions below.

Correctness.

Definition 4 (Correctness)**.** *Suppose* $\{\mathsf{FE}_i\}_{i\in[n]}$ *are correct FE candidates. We say that an FE combiner is correct if for every circuit* $C : \mathcal{X}_\lambda \to \mathcal{Y}_\lambda \in \mathcal{C}_\lambda$*, and message* $m \in \mathcal{X}_\lambda$ *it holds that:*

$$\Pr\left[\begin{array}{c} \mathsf{MSK} \leftarrow \mathsf{FEComb.Setup}(1^\lambda, \{\mathsf{FE}_i\}_{i\in[n]}) \\ \mathsf{CT} \leftarrow \mathsf{FEComb.Enc}(\mathsf{MSK}, \{\mathsf{FE}_i\}_{i\in[n]}, m) \\ \mathsf{SK}_C \leftarrow \mathsf{FEComb.Keygen}(\mathsf{MSK}, \{\mathsf{FE}_i\}_{i\in[n]}, C) \\ C(m) \leftarrow \mathsf{FEComb.Dec}(\{\mathsf{FE}_i\}_{i\in[n]}, \mathsf{SK}_C, \mathsf{CT}) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda),$$

where the probability is taken over the coins of the algorithms and $\mathsf{negl}(\lambda)$ *is a negligible function in* λ*.*

Polynomial Slowdown.

Definition 5 (Polynomial Slowdown)**.** *An FE combiner* FEComb *satisfies polynomial slowdown if on all inputs, the running times of* $\mathsf{FEComb.Setup}, \mathsf{FEComb.Enc}, \mathsf{FEComb.Keygen}$*, and* $\mathsf{FEComb.Dec}$ *are at most* $\mathsf{poly}(\lambda, n)$*, where* n *is the number of FE candidates that are being combined.*

IND-Security.

Definition 6 (IND-Secure FE Combiner)**.** *An FE combiner* FEComb *is selectively secure if for any set* $\{\mathsf{FE}_i\}_{i\in[n]}$ *of correct FE candidates, it satisfies Definition 15, where the descriptions of* $\{\mathsf{FE}_i\}_{i\in[n]}$ *are public and implicit in all invocations of the algorithms of* FEComb*,* if at least one *of the FE candidates* $\mathsf{FE}_1, \ldots, \mathsf{FE}_n$ *also satisfies Definition 15.*

Note that *Definition* 15 is the IND-security definition for FE. Unless otherwise specified, when we say a *secure* FE combiner, we refer to one that satisfies IND-security.

Simulation Security. Similarly to FE candidates, we can also consider a different notion of security called (single-key) simulation security.

Definition 7. *An FE combiner* FEComb *is single-key simulation secure if for any set* $\{\mathsf{FE}_i\}_{i\in[n]}$ *of correct FE candidates, it satisfies Definition 16, where the descriptions of* $\{\mathsf{FE}_i\}_{i\in[n]}$ *are public and implicit in all invocations of the algorithms of* FEComb*,* if at least one *of the FE candidates* $\mathsf{FE}_1, \ldots, \mathsf{FE}_n$ *also satisfies Definition 16.*

Note that *Definition* 16 is the simulation security definition for FE.

Succinctness. Similarly to FE candidates, we can also define the notion of a succinct FE combiner. An FE combiner is not required to satisfy succinctness, but we will utilize a succinct FE combiner when construction low communication MPC (Section 4 and Section 5).

Definition 8. *An FE combiner* $\mathsf{FEComb} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ *for a circuit class* $\mathcal{C}$ *containing circuits of depth at most* d *is succinct if for every set of succinct FE candidates* $\mathsf{FE}_1, \ldots, \mathsf{FE}_n$*, the following holds: For any circuit* $C \in \mathcal{C}$*,*

- *Let* $\mathsf{MSK} \leftarrow \mathsf{FEComb.Setup}(1^\lambda, \{\mathsf{FE}_i\}_{i\in[n]})$*. The size of the circuit* $\mathsf{FEComb.Enc}(\mathsf{MSK}, \cdot) \le \mathsf{poly}(n, d, \lambda)$ *for some polynomial* poly*.*
- *The function key* $\mathsf{SK}_C \leftarrow \mathsf{FEComb.KeyGen}(\mathsf{MSK}, C)$ *is of the form* (C, aux) *where* $|\mathsf{aux}| \le \mathsf{poly}(n, d, \lambda)$ *for some polynomial* poly*.*

Robust FE Combiners and Universal FE.

Remark 2. *We also define the notion of a robust FE combiner. An FE combiner* FEComb *is robust if it is an FE combiner that satisfies the three properties (correctness, polynomial slowdown, and security) associated with an FE combiner when given any set of FE candidates* $\{\mathsf{FE}_i\}_{i\in[n]}$*, provided that one is a correct and secure FE candidate. No restriction is placed on the other FE candidates. In particular, they need not satisfy correctness at all.*

Robust FE combiners can be used to build a universal functional encryption scheme defined below.

Definition 9 (T-Universal Functional Encryption)**.** *We say that an explicit Turing machine* $\Pi_{\mathsf{univ}} = (\Pi_{\mathsf{univ}}.\mathsf{Setup}, \Pi_{\mathsf{univ}}.\mathsf{Enc}, \Pi_{\mathsf{univ}}.\mathsf{KeyGen}, \Pi_{\mathsf{univ}}.\mathsf{Dec})$ *is a universal functional encryption scheme parametrized by* T *if* Π_{univ} *is a correct and secure FE scheme assuming the existence a correct and secure FE scheme with runtime* $< T$*.*

4 Succinct Single-Key Simulation Secure Decomposable FE Combiner

In this section, we define and construct a succinct single-key simulation secure decomposable FE combiner ($\mathsf{DFEComb}$ for short) that will be useful later for our communication-efficient MPC result. Such a combiner is an FE combiner that satisfies succinctness, the single-key simulation notion of security (Definition 7), and the following decomposability property.

Decomposability.

Definition 10 (Decomposability)**.** *An FE combiner* $\mathsf{FEComb} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ *is said to be decomposable if* Setup *runs* $\mathsf{MSK}_i \leftarrow \mathsf{FE}_i.\mathsf{Setup}(1^\lambda)$ *for each FE candidate* FE_i *and outputs* $\mathsf{MSK} = \{\mathsf{MSK}_i\}_{i\in[n]}$ *and* KeyGen *operates according to the following two steps.*

1. *It first runs a deterministic subroutine* $\mathsf{Partition}$ *that, on input a circuit* C *and the number of candidates* 1^n*, outputs* $(C_1, \ldots, C_n)$*, where each* C_i *is a circuit of depth polynomial in the depth of* C*.*

2. *It then runs* $\mathsf{SK}_i \leftarrow \mathsf{FE}_i.\mathsf{KeyGen}(\mathsf{MSK}_i, C_i)$ *and outputs* $\mathsf{SK}_C = \{\mathsf{SK}_i\}_{i\in[n]}$ *That is, it generates a functional key for* C_i *using the* i*th candidate and outputs the union of all these functional keys as the function key for* C*.*

When dealing with decomposable FE combiners, we will often make reference to the algorithm $\mathsf{Partition}$ used in the first step of KeyGen. We will include this algorithm as an additional algorithm for such combiners.

Such a notion of decomposability is natural when we consider the connection between FE combiners and MPC. In particular, in MPC, each party generates messages and broadcasts them to the other parties. When relating FE combiners to MPC, it will be useful to associate each FE candidate to a party in the MPC protocol. Decomposability of the FE combiner intuitively allows each party to generate a functional key corresponding to their FE candidate and broadcast it to the other parties. Once each party has acquired all the functional keys from the other parties, the decomposability property ensures that each party possesses the functional key for the FE combiner.

4.1 Construction of a DFE Combiner from LWE

In this section, we give our construction of a succinct single-key simulation secure decomposable FE combiner $\mathsf{DFEComb}$ from LWE. For the construction let, $\mathsf{TFHE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{PartDec}, \mathsf{FinDec})$ denote a threshold homomorphic encryption scheme. Let $\mathsf{FE}_1, \ldots, \mathsf{FE}_n$ be n FE candidates. For simplicity of the exposition, all the algorithms of $\mathsf{DFEComb}$ implicitly get the description of the candidates as input. We show the following result.

Theorem 5. *Assuming* TFHE *is a secure threshold (leveled) homomorphic encryption scheme, there exists a succinct single-key simulation secure decomposable FE combiner.*

Corollary 1. *Assuming LWE, there exists a succinct single-key simulation secure decomposable FE combiner.*

Construction. Since we are constructing a *succinct* FE combiner, the combiner and FE candidates will be instantiated for circuit classes with bounded depth. To make this clear, we input the depth of the circuit class to the setup algorithm in the construction. Additionally, we include $\mathsf{Partition}$ in the list of algorithms for $\mathsf{DFEComb}$ as we will reference it later.

- $\mathsf{DFEComb.Setup}(1^\lambda, 1^d, 1^n)$: It runs $\mathsf{FE}_i.\mathsf{Setup}(1^\lambda, 1^{d'}) \to \mathsf{MSK}_i$ for all $i \in [n]$ and outputs $\mathsf{MSK} = \{\mathsf{MSK}_i\}_{i\in[n]}$.

  Here, d' is the depth of the circuit $\mathsf{TFHE.PartDec}(\cdot, \mathsf{TFHE.Eval}(\cdot,\cdot))$ that evaluates a TFHE ciphertext using a circuit of depth d, and then computes a partial decryption. Note that, $d' = \mathsf{poly}(\lambda, d)$ due to compactness of TFHE.

- $\mathsf{DFEComb.Enc}(\mathsf{MSK}, m)$: It executes the setup of TFHE; $\mathsf{TFHE.Setup}(1^\lambda, 1^d, 1^n) \to (\mathsf{fpk}, \mathsf{fsk}_1, \ldots, \mathsf{fsk}_n)$. Then, it proceeds as follows:
  - Compute $\mathsf{TFHE.Enc}(\mathsf{fpk}, m) \to \mathsf{CT}_{\mathsf{fhe}}$
  - For all $i \in [n]$, sample $r_i \xleftarrow{\$} \{0,1\}^{\ell_{\mathsf{fhe}}}$. Here, ℓ_{fhe} is the length of the randomness required by the $\mathsf{PartDec}$ algorithm.
  - Compute $\mathsf{CT}_i \leftarrow \mathsf{FE}_i.\mathsf{Enc}\left(\mathsf{MSK}_i, (\mathsf{fpk}, \mathsf{CT}_{\mathsf{fhe}}, \mathsf{fsk}_i, r_i)\right)$.
  - Output $\mathsf{CT} = \{\mathsf{CT}_i\}_{i\in[n]}$.

- $\mathsf{DFEComb.Partition}(C)$: Consider the circuit F_C that takes as input four strings from implicit domains of the form $(\mathsf{fpk}', \mathsf{CT}'_{\mathsf{fhe}}, \mathsf{SK}', r')$ and computes

$$p \leftarrow \mathsf{TFHE.PartDec}\left(\mathsf{SK}', \mathsf{TFHE.Eval}\left(\mathsf{fpk}', \mathsf{CT}'_{\mathsf{fhe}}, C\right); r'\right).$$

  The partition function outputs $(F_C, \ldots, F_C)$.

- $\mathsf{DFEComb.Keygen}(\mathsf{MSK}, C)$: The key generation algorithm first computes $(C_1, \ldots, C_n) \leftarrow \mathsf{DFEComb.Partition}(C)$. Then, it computes $(C_i, \mathsf{SK}_i) \leftarrow \mathsf{FE}_i.\mathsf{KeyGen}(\mathsf{MSK}_i, C_i)$ for $i \in [n]$. It outputs $\mathsf{SK} = (C, \{\mathsf{SK}_i\}_{i\in[n]})$.

- $\mathsf{DFEComb.Dec}(\mathsf{SK}, \mathsf{CT})$: The decryption algorithm first parses SK as $(C, \{\mathsf{SK}_i\}_{i\in[n]})$ and CT as $\{\mathsf{CT}_i\}_{i\in[n]}$. Then, it computes $p_i \leftarrow \mathsf{FE}_i.\mathsf{Dec}(\mathsf{SK}_i, \mathsf{CT}_i)$. Finally, it outputs $\mathsf{TFHE.FinDec}(p_1, \ldots, p_n)$.

Lemma 2. *The FE combiner construction above satisfies correctness, polynomial slowdown, and decomposability.*

The correctness and efficiency of the combiner follow immediately from the construction and the correctness and efficiency of the FE candidates and TFHE. Decomposability follows immediately from the construction.

Lemma 3. *The FE combiner construction above satisfies succinctness.*

Let's denote $|m|$ to be ℓ_{in}. Let's first bound the size of the encryption circuit. The ciphertext contains encryptions of $\mathsf{fpk}, \mathsf{CT}_{\mathsf{fhe}}, \mathsf{fsk}_i, r_i$ using the candidate i. If each candidate FE_i satisfies the efficiency requirement, then the size of the circuit computing each ciphertext is just $\mathsf{poly}_i(\lambda, \ell_{\mathsf{in}}, d, \ell_{\mathsf{out}})$ for some polynomial poly_i as it only depends on the depth of $\mathsf{TFHE.PartDec}(\mathsf{TFHE.Eval}(\cdot, C); \cdot)$, the length of $\mathsf{CT}_{\mathsf{fhe}}, \mathsf{fpk}, \mathsf{fsk}_i, r_i$ and the length of the output. The size of the circuit that computes $\mathsf{TFHE.Enc}$ is bounded by some polynomial $\mathsf{poly}(\lambda, \ell_{\mathsf{in}}, d)$ due to the compactness property of the TFHE scheme. Thus, the total size of the circuit is bounded by some polynomial $\mathsf{poly}(n, \lambda, d, \ell_{\mathsf{in}})$. Furthermore, $|\{\mathsf{SK}_i\}_{i\in[n]}| < \mathsf{poly}(n, \lambda, d, \ell_{\mathsf{in}}, \ell_{\mathsf{out}})$ since each of the underlying FE candidates are succinct.

Theorem 6. *The FE combiner construction above is single-key simulation secure.*

Input: Master secret key $\mathsf{MSK} = (\mathsf{MSK}_1, .., \mathsf{MSK}_n)$, functional key $\mathsf{SK}_C = \{\mathsf{SK}_i\}_{i\in[n]}$, circuit C, value $C(m)$, set of insecure candidates $I = [n] \setminus i^*$.

- Let $I = [n] \setminus \{i^*\}$.
- Run $\mathsf{TFHE.Setup}(1^\lambda, 1^d, 1^n) \to (\mathsf{fpk}, \mathsf{fsk}_1, \ldots, \mathsf{fsk}_n)$
- Compute $\mathsf{TFHE.Enc}(\mathsf{fpk}, 0) \to \mathsf{CT}_{\mathsf{fhe}}$
- For all $i \in I$, sample $r_i \xleftarrow{\$} \{0,1\}^{\ell_{\mathsf{fhe}}}$. Here, ℓ_{fhe} is the length of the randomness required by $\mathsf{PartDec}$ algorithm.
- Let $\mathsf{TFHE.PartDec}(\mathsf{TFHE.Eval}(C, \mathsf{CT}_{\mathsf{fhe}}), \{\mathsf{fsk}_i\}_{i\in I}, C(m)) \to p_{i^*}$
- Compute $\mathsf{CT}_i \leftarrow \mathsf{FE}_i.\mathsf{Enc}\left(\mathsf{MSK}_i, (\mathsf{fpk}, \mathsf{CT}_{\mathsf{fhe}}, \mathsf{fsk}_i, r_i)\right)$ for all $i \in I$.
- Compute $\mathsf{CT}_{i^*} \leftarrow \mathsf{FE}_{i^*}.\mathsf{Sim}(\mathsf{MSK}_{i^*}, C, \mathsf{SK}_{i^*}, p_{i^*})$
- Output $\mathsf{CT} = \{\mathsf{CT}_i\}_{i\in[n]}$.

Figure 1: Simulator of $\mathsf{DFEComb}$

This is the last step towards proving Theorem 5. We give a description of the simulator in Figure 1.

We now prove security via a sequence of hybrid experiments by arguing that every pair of consecutive hybrids is computationally indistinguishable. The first hybrid, Hyb_0, corresponds to the real experiment, while the last hybrid, Hyb_3, corresponds to the simulated experiment. We will use red, underlined text to denote the differences between consecutive hybrids.

Hyb_0 : This hybrid is the real experiment. Namely,

1. For every $i \in [n]$, run $\mathsf{FE}_i.\mathsf{Setup}(1^\lambda, 1^{d'}) \to \mathsf{MSK}_i$
2. Compute $\mathsf{DFEComb.Keygen}(\mathsf{MSK}, C) \to \mathsf{SK}_C = (\mathsf{SK}_1, \ldots, \mathsf{SK}_n)$
3. $\mathcal{A}_1(\{\mathsf{MSK}_i\}_{i\in I}, \mathsf{SK}_C) \to (m, z_1)$
4. Run $\mathsf{TFHE.Setup}(1^\lambda, 1^d, 1^n) \to (\mathsf{fpk}, \mathsf{fsk}_1, \ldots, \mathsf{fsk}_n)$
5. Compute $\mathsf{TFHE.Enc}(\mathsf{fpk}, m) \to \mathsf{CT}_{\mathsf{fhe}}$
6. For all $i \in [n]$, sample $r_i \xleftarrow{\$} \{0,1\}^{\ell_{\mathsf{fhe}}}$. Here, ℓ_{fhe} is the length of the randomness required by $\mathsf{PartDec}$ algorithm.
7. Compute $\mathsf{CT}_i \leftarrow \mathsf{FE}_i.\mathsf{Enc}\left(\mathsf{MSK}_i, (\mathsf{fpk}, \mathsf{CT}_{\mathsf{fhe}}, \mathsf{fsk}_i, r_i)\right)$.
8. Output $\mathsf{CT} = \{\mathsf{CT}_i\}_{i\in[n]}$
9. $\mathcal{A}_2(\mathsf{CT}, \mathsf{SK}_C, z_1) \to \alpha$
10. Output α

Hyb_1 : This hybrid is exactly the same as the previous one except that now, the output ciphertext of the secure candidate FE_1 (without loss of generality, assume that FE_1 is a secure

candidate and $I = [2, n]$ for the rest of the hybrids) is simulated using the correct TFHE partial decryption p_1 (defined below).

1. For every $i \in [n]$, run $\mathsf{FE}_i.\mathsf{Setup}(1^\lambda, 1^{d'}) \to \mathsf{MSK}_i$
2. Compute $\mathsf{DFEComb.Keygen}(\mathsf{MSK}, C) \to \mathsf{SK}_C = (\mathsf{SK}_1, \ldots, \mathsf{SK}_n)$
3. $\mathcal{A}_1(\{\mathsf{MSK}_i\}_{i \in I}, \mathsf{SK}_C) \to (m, z_1)$
4. Run $\mathsf{TFHE.Setup}(1^\lambda, 1^d, 1^n) \to (\mathsf{fpk}, \mathsf{fsk}_1, \ldots, \mathsf{fsk}_n)$
5. Compute $\mathsf{TFHE.Enc}(\mathsf{fpk}, m) \to \mathsf{CT}_{\mathsf{fhe}}$
6. For all $i \in [n]$, sample $r_i \xleftarrow{\$} \{0,1\}^{\ell_{\mathsf{fhe}}}$. Here, ℓ_{fhe} is the length of the randomness required by $\mathsf{PartDec}$ algorithm.
7. Let $\mathsf{TFHE.PartDec}(\mathsf{TFHE.Eval}(\mathsf{CT}_{\mathsf{fhe}}, C), \mathsf{SK}_1; r_1) \to p_1$
8. Compute $\mathsf{CT}_i \leftarrow \mathsf{FE}_i.\mathsf{Enc}\left(\mathsf{MSK}_i, (\mathsf{fpk}, \mathsf{CT}_{\mathsf{fhe}}, \mathsf{fsk}_i, r_i)\right)$ for all $i \in [2, n]$.
9. Compute $\mathsf{CT}_1 \leftarrow \mathsf{FE}_1.\mathsf{Sim}(\mathsf{MSK}_1, C, \mathsf{SK}_1, p_1)$
10. Output $\mathsf{CT} = \{\mathsf{CT}_i\}_{i \in [n]}$
11. $\mathcal{A}_2(\mathsf{CT}, \mathsf{SK}_C, z_1) \to \alpha$
12. Output α

Hyb_2 : This hybrid is exactly the same as the previous one except that now, the TFHE partial decryption is simulated using secret keys $\mathsf{fsk}_2, \ldots, \mathsf{fsk}_n$

1. For every $i \in [n]$, run $\mathsf{FE}_i.\mathsf{Setup}(1^\lambda, 1^{d'}) \to \mathsf{MSK}_i$
2. Compute $\mathsf{DFEComb.Keygen}(\mathsf{MSK}, C) \to \mathsf{SK}_C = (\mathsf{SK}_1, \ldots, \mathsf{SK}_n)$
3. $\mathcal{A}_1(\{\mathsf{MSK}_i\}_{i \in I}, \mathsf{SK}_C) \to (m, z_1)$
4. Run $\mathsf{TFHE.Setup}(1^\lambda, 1^d, 1^n) \to (\mathsf{fpk}, \mathsf{fsk}_1, \ldots, \mathsf{fsk}_n)$
5. Compute $\mathsf{TFHE.Enc}(\mathsf{fpk}, m) \to \mathsf{CT}_{\mathsf{fhe}}$
6. For all $i \in [n]$, sample $r_i \xleftarrow{\$} \{0,1\}^{\ell_{\mathsf{fhe}}}$. Here, ℓ_{fhe} is the length of the randomness required by $\mathsf{PartDec}$ algorithm.
7. Let $\mathsf{TFHE.Sim}(\mathsf{TFHE.Eval}(C, \mathsf{CT}_{\mathsf{fhe}}), \{\mathsf{fsk}_i\}_{i \in [2,n]}, C(m)) \to p_1$
8. Compute $\mathsf{CT}_i \leftarrow \mathsf{FE}_i.\mathsf{Enc}\left(\mathsf{MSK}_i, (\mathsf{fpk}, \mathsf{CT}_{\mathsf{fhe}}, \mathsf{fsk}_i, r_i)\right)$ for all $i \in [2, n]$.
9. Compute $\mathsf{CT}_1 \leftarrow \mathsf{FE}_1.\mathsf{Sim}(\mathsf{MSK}_1, C, \mathsf{SK}_1, p_1)$
10. Output $\mathsf{CT} = \{\mathsf{CT}_i\}_{i \in [n]}$
11. $\mathcal{A}_2(\mathsf{CT}, \mathsf{SK}_C, z_1) \to \alpha$
12. Output α

Hyb_3 : This hybrid is exactly the same as the previous one except that now, $\mathsf{CT}_{\mathsf{fhe}}$ is set as encryption of 0.

1. For every $i \in [n]$, run $\mathsf{FE}_i.\mathsf{Setup}(1^\lambda, 1^{d'}) \to \mathsf{MSK}_i$

2. Compute $\mathsf{DFEComb.Keygen}(\mathsf{MSK}, C) \to \mathsf{SK}_C = (\mathsf{SK}_1, \ldots, \mathsf{SK}_n)$
3. $\mathcal{A}_1(\{\mathsf{MSK}_i\}_{i\in I}, \mathsf{SK}_C) \to (m, z_1)$
4. Run $\mathsf{TFHE.Setup}(1^\lambda, 1^d, 1^n) \to (\mathsf{fpk}, \mathsf{fsk}_1, \ldots, \mathsf{fsk}_n)$
5. Compute $\mathsf{TFHE.Enc}(\mathsf{fpk}, 0) \to \mathsf{CT}_{\mathsf{fhe}}$
6. For all $i \in [n]$, sample $r_i \xleftarrow{\$} \{0,1\}^{\ell_{\mathsf{fhe}}}$. Here, ℓ_{fhe} is the length of the randomness required by $\mathsf{PartDec}$ algorithm.
7. Let $\mathsf{TFHE.Sim}(\mathsf{TFHE.Eval}(C, \mathsf{CT}_{\mathsf{fhe}}), \{\mathsf{fsk}_i\}_{i\in[2,n]}, C(m)) \to p_1$
8. Compute $\mathsf{CT}_i \leftarrow \mathsf{FE}_i.\mathsf{Enc}\left(\mathsf{MSK}_i, (\mathsf{fpk}, \mathsf{CT}_{\mathsf{fhe}}, \mathsf{fsk}_i, r_i)\right)$ for all $i \in [2, n]$.
9. Compute $\mathsf{CT}_1 \leftarrow \mathsf{FE}_1.\mathsf{Sim}(\mathsf{MSK}_1, C, \mathsf{SK}_1, p_1)$
10. Output $\mathsf{CT} = \{\mathsf{CT}_i\}_{i\in[n]}$
11. $\mathcal{A}_2(\mathsf{CT}, \mathsf{SK}_C, z_1) \to \alpha$
12. Output α

Lemma 4. *Assuming* FE_1 *is single-key simulation-secure,* Hyb_0 *is indistinguishable to* Hyb_1.

Proof. The only difference between the two hybrids is the way CT_1 is generated. In Hyb_0, it is generated honestly while in Hyb_1, it is simulated using the correct output p_1. We sketch the reduction here. The reduction generates TFHE parameters and the functional encryption keys for all candidates in $[2, n]$. Then, it generates the value p_1 correctly and gives p_1 to the challenger. The challenger either sends a simulated FE_1 ciphertext or the actual ciphertext. This ciphertext is forwarded to the adversary. The reduction outputs whatever the adversary outputs.

□

Lemma 5. *Assuming* TFHE *is a secure threshold (leveled) homomorphic encryption scheme,* Hyb_1 *is indistinguishable to* Hyb_2.

Proof. The only difference between the two hybrids is the way p_1 is generated. In Hyb_1, it is generated honestly while in Hyb_2, it is simulated using the secret keys $\mathsf{fsk}_2, \ldots, \mathsf{fsk}_n$ and the output $C(m)$. We sketch the reduction here. The adversary sends $[2, n]$ to the reduction. The reduction sends it to the challenger and gets $(\mathsf{fpk}, \mathsf{fsk}_2, \ldots, \mathsf{fsk}_n)$ from the challenger. Then, the reduction generates $\mathsf{CT}_{\mathsf{fhe}}$ using the message m sent by the adversary. The reduction also generates FE_i keys and interacts with the adversary using this information. It sends $C(m)$ to the challenger. It either receives the actual p_1 or a simulated p_1 from the challenger. This response is then used to interact with the adversary. It outputs whatever the adversary outputs.

□

Lemma 6. *Assuming* TFHE *is a secure threshold (leveled) homomorphic encryption scheme,* Hyb_2 *is indistinguishable to* Hyb_3.

Proof. The only difference between the two hybrids is the way $\mathsf{CT}_{\mathsf{fhe}}$ is generated. In Hyb_2, it is generated honestly while in Hyb_3, it is generated as an encryption of 0. Note that both hybrids are independent of fsk_1. Indistinguishability follows from the semantic security of TFHE. □

Instantiation: We have given a construction of a succinct single-key simulation secure decomposable FE combiner. We note that we can potentially instantiate all the candidates using the same scheme.

In particular, the FE construction in [GKP+13] can be made to satisfy these requirements. The candidate in [GKP+13] is based on any leveled FHE scheme, garbling scheme, and attribute-based encryption (ABE) scheme for circuits. In order for the FE candidate to satisfy succinctness, the underlying ABE scheme must also be succinct. This can be done by instantiating the ABE scheme with [BGG+14, GVW15]. and the leveled FHE scheme with [GSW13, BGV14] as described in the technical overview. All these results are based on LWE. Therefore, we arrive at the following lemma.

Lemma 7. *The FE construction in [GKP+13] is succinct according to Definition 2.*

Combining the above lemma and Theorem 5, we get the following corollary:

Corollary 2. *Assuming LWE, there exists a single-key simulation secure succinct FE scheme.*

Remark: At first glance, it might seem weird that we want to instantiate all the candidates using the same scheme, seemingly defeating the purpose of an FE combiner. However, looking ahead to the application to MPC, we associate each FE candidate with a party in the MPC protocol. There, the adversary might corrupt an arbitrary set of parties, which would translate to insecure FE candidates. Therefore, by instantiating all the candidates with the same (secure) scheme, we guarantee that the honest parties' inputs remain secure. More details on this are provided in the next section.

5 Round Optimal MPC with Depth-Proportional Communication from an FE Combiner

In this section, using any succinct single-key simulation secure decomposable FE combiner (see Section 4), we show how to compile any two round semi-honest secure MPC protocol into one where the communication complexity is proportional only to the depth of the circuit being evaluated.

Let $\mathsf{Comm.Compl}(\pi)$ denote the communication complexity of any protocol π. Let λ denote the security parameter, n denote the number of parties, and ℓ denote the size of the input to each party. Formally, we show the following theorem:

Theorem 7. *Assuming the existence of*

- *A succinct single-key single-ciphertext simulation secure decomposable FE combiner (AND)*
- *Succinct FE candidates (AND)*
- *A two round semi-honest MPC in the plain model (that may not be communication efficient) that is secure against up to all but one corruption,*

there exists a two round semi-honest MPC protocol π in the plain model that is secure against up to all but one corruption for any boolean circuit C, where the communication complexity of the protocol π is independent of the size of the circuit. That is, $\mathsf{Comm.Compl}(\pi) = \mathsf{poly}(\mathsf{Depth}(C), n, \ell, \lambda)$.

We know how to construct a succinct single-key single-key simulation secure decomposable FE combiner based on the learning with errors (LWE) assumption (see Section 4). Further, from Lemma 7, we know that the construction in [GKP+13] is a succinct FE candidate. Also, two round semi-honest MPC protocols secure against up to all but one corruption can be based

on the LWE assumption [GS18, BL18, PVW08][4]. Instantiating the primitives in the above theorem, we get the following corollary:

Corollary 3. *Assuming LWE, there exists a two round semi-honest MPC protocol π in the plain model that is secure against up to all but one corruption for any boolean circuit C with* $\mathsf{Comm.Compl}(\pi) = \mathsf{poly}(\mathsf{Depth}(C), n, \ell, \lambda)$.

5.1 Construction

Notation:

- Consider n parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$ with inputs $x_1, \ldots, x_n$, respectively, who wish to evaluate a boolean circuit C on their joint inputs. Let λ denote the security parameter and without loss of generality, let's assume $|x_i| = \lambda$ for all $i \in [n]$. Also, let's denote the randomness of each party P_i as $r_i = (r_i^{\mathsf{Setup}}, r_i^{\mathsf{Enc}}, r_i^{\mathsf{SH}}, r_i^{\mathsf{KeyGen}})$.
- Let $\mathsf{DFEComb} = (\mathsf{DFEComb.Setup}, \mathsf{DFEComb.Enc}, \mathsf{DFEComb.Keygen}, \mathsf{DFEComb.Dec}, \mathsf{DFEComb.Partition})$ be a succinct single-key simulation secure decomposable FE combiner (see Section 4) for n FE candidates $\mathsf{FE}_1, \ldots, \mathsf{FE}_n$.
- Let π^{SH} be a two round semi-honest secure MPC protocol (not necessarily communication efficient). Let $(\pi^{\mathsf{SH}}.\mathsf{Round}_1, \pi^{\mathsf{SH}}.\mathsf{Round}_2)$ denote the algorithms used by any party to compute the messages in each of the two rounds and $\pi^{\mathsf{SH}}.\mathsf{Out}$ denote the algorithm to compute the final output. Further, let $\pi^{\mathsf{SH}}.\mathsf{Sim} = (\pi^{\mathsf{SH}}.\mathsf{Sim}_1, \pi^{\mathsf{SH}}.\mathsf{Sim}_2)$ denote the simulator for this protocol - that is, $\pi^{\mathsf{SH}}.\mathsf{Sim}_i$ is the simulator's algorithm to compute the i^{th} round's messages.

Protocol. We now describe the construction of our protocol π with depth-proportional communication complexity.

- **Round 1:** Each party P_i does the following:
  1. Generate $\mathsf{MSK}_i \leftarrow \mathsf{FE}_i.\mathsf{Setup}(1^\lambda)$ using randomness r_i^{Setup}.
  2. Compute $(C_1, \ldots, C_n) \leftarrow \mathsf{DFEComb.Partition}(1^\lambda, C)$.
  3. Compute $\mathsf{SK}_i = \mathsf{FE}_i.\mathsf{KeyGen}(\mathsf{MSK}_i, C_i)$ using randomness r_i^{KeyGen}.
  4. Participate in an execution of protocol π^{SH} with the remaining $(n-1)$ parties using input $y_i = (x_i, \mathsf{MSK}_i, r_i^{\mathsf{Enc}})$ and randomness r_i^{SH} to compute the deterministic circuit C_{CT} defined in Figure 2. That is, compute the first round message $\mathsf{msg}_{1,i} \leftarrow \pi^{\mathsf{SH}}.\mathsf{Round}_1(y_i; r_i^{\mathsf{SH}})$.
  5. Output $(\mathsf{msg}_{1,i}, \mathsf{SK}_i)$.
- **Round 2:** Each party P_i does the following:
  1. Let τ_1 denote the transcript of protocol π^{SH} after round 1.
  2. Compute the second round message $\mathsf{msg}_{2,i} \leftarrow \pi^{\mathsf{SH}}.\mathsf{Round}_2(y_i, \tau_1; r_i^{\mathsf{SH}})$ where $y_i = (x_i, \mathsf{MSK}_i, r_i^{\mathsf{Enc}})$.
  3. Output $(\mathsf{msg}_{2,i})$.
- **Output Computation:** Each party P_i does the following:
  1. Let τ_2 denote the transcript of protocol π^{SH} after round 2.

[4] [GS18, BL18] showed how to construct two round semi-honest MPC in the plain model from any two round semi-honest OT in the plain model and [PVW08] show that the latter can be constructed from LWE.

2. Compute the output of π^{SH} as $\mathsf{CT} \leftarrow \pi^{\mathsf{SH}}.\mathsf{Out}(y_i, \tau_2; r_i^{\mathsf{SH}})$.
3. Let $\mathsf{SK}_C = (\mathsf{SK}_1, \ldots, \mathsf{SK}_n)$.
4. Output $\mathsf{DFEComb.Dec}(\mathsf{SK}_C, \mathsf{CT})$.

Input: $\{(x_i, \mathsf{MSK}_i, r_i^{\mathsf{Enc}})\}_{i=1}^n$

- Let $\mathsf{MSK} = (\mathsf{MSK}_1, \ldots, \mathsf{MSK}_n)$, $x = (x_1, \ldots, x_n)$ and $r = (r_1^{\mathsf{Enc}}, \ldots, r_n^{\mathsf{Enc}})$.
- Output $\mathsf{DFEComb.Enc}(\mathsf{MSK}, x)$ using randomness r.

Figure 2: Circuit C_{CT}

Correctness and Efficiency: Correctness follows immediately from the construction. In particular, at the end of the protocol, each party possesses CT, an encryption of $x = (x_1, \ldots, x_n)$ under the FE combiner, and SK_C, the function key for C. This ciphertext can then be decrypted using SK_C to yield $C(x)$, as desired.

Now, let's analyze the communication complexity of the protocol. First, observe that for each circuit C that is of depth d and outputs a single bit, the size of the output length of C_i for all $i \in [n]$ is at most $\mathsf{poly}(\lambda, d)$ bits where $(C_1, \ldots, C_n) \leftarrow \mathsf{DFEComb.Partition}(1^\lambda, C)$. This is due to the compactness of the FHE scheme.

Then, from Section 4, we know that $|\mathsf{SK}_i| = p_1(d, n, \lambda)$ and $|\mathsf{CT}| = p_2(d, n, \lambda)$ where p_1, p_2 are both polynomials. Recall that CT is the ciphertext that is the output of the protocol π^{SH} (computed during decryption). Then, for the protocol π^{SH} recall that the input is $y_i = (x_i, \mathsf{MSK}_i, r_i^{\mathsf{Enc}})$ and so $|y_i| = p_3(\lambda, d)$ for some polynomial p_3. Therefore, for each party P_i, $|\mathsf{msg}_{1,i}| = p_4(d, n, \lambda)$ and $|\mathsf{msg}_{2,i}| = p_5(d, n, \lambda)$ for some polynomials p_4, p_5.

Therefore, in our two round protocol π, in each round the size of the message sent by any party is $\mathsf{poly}(n, d, \lambda)$. Thus, $\mathsf{Comm.Compl}(\pi) = \mathsf{poly}(n, d, \lambda)$.

For circuits that output multi-bit strings, the communication complexity of our MPC protocol π is bounded by $\mathsf{poly}(n, d, \lambda) \cdot \ell_{\mathsf{out}}$ where ℓ_{out} is the output length of the circuit.

5.2 Security Proof

We will first describe the simulator and then show that the real and ideal worlds are indistinguishable. Suppose the adversary corrupts a set of k parties (denoted by Corr) with $k < n$. At a very high level, the simulator does the following: simulate the underlying semi-honest MPC protocol whose output - the FE combiner ciphertext - is computed using the simulator of the FE combiner. Below, we formally describe the simulator.

Description of the Simulator: The simulator Sim gets as input $(\{m_i, r_i\}_{\mathsf{P}_i \in \mathsf{Corr}}, C(\{m_i\}_{i \in [n]}), C)$ and does the following:

- **<u>Round 1:</u>**
  1. Compute $(C_1, \ldots, C_n) \leftarrow \mathsf{DFEComb.Partition}(1^\lambda, C)$.
  2. For each corrupt party P_i, parse $r_i = (r_i^{\mathsf{Setup}}, r_i^{\mathsf{Enc}}, r_i^{\mathsf{SH}}, r_i^{\mathsf{KeyGen}})$. Generate $\mathsf{MSK}_i \leftarrow \mathsf{FE}_i.\mathsf{Setup}(1^\lambda)$ and $\mathsf{SK}_i = \mathsf{FE}_i.\mathsf{KeyGen}(\mathsf{MSK}_i, C_i)$ using randomness $r_i^{\mathsf{Setup}}, r_i^{\mathsf{KeyGen}}$ respectively.

3. For each honest party P_j, compute $\mathsf{MSK}_j \leftarrow \mathsf{FE}_j.\mathsf{Setup}(1^\lambda)$ and $\mathsf{SK}_j = \mathsf{FE}_j.\mathsf{KeyGen}(\mathsf{MSK}_j, C_j)$ using uniformly generated randomness.
4. Pick one honest party $\mathsf{P}_{j^*} \notin \mathsf{Corr}$ at random amongst the set of honest parties and let I denote the set of indices of all parties except P_{j^*}.
5. Compute $\mathsf{CT}^* = \mathsf{DFEComb.Sim}(\{\mathsf{MSK}_i\}_{i\in[n]}, \{\mathsf{SK}_i\}_{i\in[n]}, C, C(\{m_i\}_{i\in[n]}), I)$.
6. Compute $\mathsf{msg}_1 \leftarrow \pi^{\mathsf{SH}}.\mathsf{Sim}_1(\mathsf{CT}^*, \{m_i, \mathsf{MSK}_i, r_i^{\mathsf{Enc}}\}_{\mathsf{P}_i\in\mathsf{Corr}}, \{r_i^{\mathsf{SH}}\}_{\mathsf{P}_i\in\mathsf{Corr}})$. where $\{r_i^{\mathsf{SH}}\}_{\mathsf{P}_i\in\mathsf{Corr}}$ is the randomness of all the parties. Here, msg_1 denotes the set of the first round messages of all the honest parties.
7. Output $(\mathsf{msg}_1, \{\mathsf{SK}_j\}_{\mathsf{P}_j\in[n]/\mathsf{Corr}})$.

- **Round 2:**
  1. Compute $\mathsf{msg}_2 \leftarrow \pi^{\mathsf{SH}}.\mathsf{Sim}_2(\mathsf{CT}^*, \{m_i, \mathsf{MSK}_i, r_i^{\mathsf{Enc}}\}_{\mathsf{P}_i\in\mathsf{Corr}}, \{r_i^{\mathsf{SH}}\}_{\mathsf{P}_i\in\mathsf{Corr}})$ as the second round message of π^{SH}. Here, msg_2 denotes the set of the first round messages of all the honest parties.
  2. Output (msg_2).

Hybrids. We will now complete the proof of Theorem 7 via a sequence of computationally indistinguishable hybrids $\mathsf{Hyb}_0, \mathsf{Hyb}_1, \mathsf{Hyb}_2$ where Hyb_0 corresponds to the real world and Hyb_2 corresponds to the ideal world.

Hyb_0 **- Real World:** In this hybrid, consider a simulator SimHyb that plays the role of the honest parties. This corresponds to the real world experiment.

Hyb_1 **- Simulate MPC:** In this hybrid, SimHyb computes the messages of π^{SH} using the simulator. That is, SimHyb does the following:

- **Round 1:**
  1. Compute $(C_1, \ldots, C_n) \leftarrow \mathsf{DFEComb.Partition}(1^\lambda, C)$.
  2. For each corrupt party P_i, parse $r_i = (r_i^{\mathsf{Setup}}, r_i^{\mathsf{Enc}}, r_i^{\mathsf{SH}}, r_i^{\mathsf{KeyGen}})$. Generate $\mathsf{MSK}_i \leftarrow \mathsf{FE}_i.\mathsf{Setup}(1^\lambda)$ and $\mathsf{SK}_i = \mathsf{FE}_i.\mathsf{KeyGen}(\mathsf{MSK}_i, C_i)$ using randomness $r_i^{\mathsf{Setup}}, r_i^{\mathsf{KeyGen}}$ respectively.
  3. For each honest party P_j, compute $\mathsf{MSK}_j \leftarrow \mathsf{FE}_j.\mathsf{Setup}(1^\lambda)$ and $\mathsf{SK}_j = \mathsf{FE}_j.\mathsf{KeyGen}(\mathsf{MSK}_j, C_j)$ using uniformly generated randomness.
  4. Compute $\mathsf{CT}^* = \mathsf{DFEComb.Enc}(\{\mathsf{MSK}_i\}_{i\in[n]}, \{m_i\}_{i\in[n]})$ using randomness $\{r_i^{\mathsf{Enc}}\}_{i\in[n]}$, which are picked uniformly at random for the honest parties.
  5. Compute $\mathsf{msg}_1 \leftarrow \pi^{\mathsf{SH}}.\mathsf{Sim}_1(\mathsf{CT}^*, \{m_i, \mathsf{MSK}_i, r_i^{\mathsf{Enc}}\}_{\mathsf{P}_i\in\mathsf{Corr}}, \{r_i^{\mathsf{SH}}\}_{\mathsf{P}_i\in\mathsf{Corr}})$. where $\{r_i^{\mathsf{SH}}\}_{\mathsf{P}_i\in\mathsf{Corr}}$ is the randomness of all the parties.
  6. Output $(\mathsf{msg}_1, \{\mathsf{SK}_j\}_{\mathsf{P}_j\in[n]/\mathsf{Corr}})$.

- **Round 2:**
  1. Compute $\mathsf{msg}_2 \leftarrow \pi^{\mathsf{SH}}.\mathsf{Sim}_2(\mathsf{CT}^*, \{m_i, \mathsf{MSK}_i, r_i^{\mathsf{Enc}}\}_{\mathsf{P}_i\in\mathsf{Corr}}, \{r_i^{\mathsf{SH}}\}_{\mathsf{P}_i\in\mathsf{Corr}})$ as the second round message of π^{SH}.
  2. Output (msg_2)

  .

Hyb_2 **- Simulate FE:** In this hybrid, SimHyb does the following differently:

- Pick one honest party P_{j^*} at random and let I denote the set of indices of all parties except P_{j^*}.
- Compute $\mathsf{CT}^* = \mathsf{DFEComb.Sim}(\{\mathsf{MSK}_i\}_{i\in[n]}, \{\mathsf{SK}_i\}_{i\in[n]}, C, C(\{m_i\}_{i\in[n]}), I)$.

This corresponds to the ideal world.

We will now show the indistinguishability of consecutive hybrids.

Lemma 8. *Assuming π^{SH} is a semi-honest MPC protocol secure against up to all but one corruption, Hyb_0 is computationally indistinguishable from Hyb_1.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that can distinguish between the two hybrids with non-negligible probability. We will use $\mathcal{A}$ to construct an adversary $\mathcal{A}_{\mathsf{SH}}$ that breaks the security of the MPC protocol π^{SH}, which is a contradiction.

The adversary $\mathcal{A}_{\mathsf{SH}}$ interacts with the adversary $\mathcal{A}$ as in Hyb_0. $\mathcal{A}_{\mathsf{SH}}$ corrupts a set of parties Corr and for each $\mathsf{P}_i \in \mathsf{Corr}$, $\mathcal{A}_{\mathsf{SH}}$ receives (m_i) from $\mathcal{A}$. $\mathcal{A}_{\mathsf{SH}}$ interacts with the challenger $\mathcal{C}_{\mathsf{SH}}$ in an MPC protocol π^{SH} with (n) parties to compute circuit C_{CT} defined in Figure 2 and corrupts the same set of parties. In this protocol, for each corrupt party P_i, $\mathcal{A}_{\mathsf{SH}}$ picks random $r_i^{\mathsf{Setup}}, r_i^{\mathsf{Enc}}, r_i^{\mathsf{KeyGen}}$ and computes $\mathsf{MSK}_i = \mathsf{FE}_i.\mathsf{Setup}(1^\lambda)$ using randomness r_i^{Setup}. Each corrupt party uses input $(m_i, \mathsf{MSK}_i, r_i^{\mathsf{Enc}})$. $\mathcal{A}_{\mathsf{SH}}$ sends the input of each corrupt party to $\mathcal{C}_{\mathsf{SH}}$ and receives its randomness r_i^{SH}. Then, in the interaction with $\mathcal{A}$, for each corrupt party P_i, $\mathcal{A}_{\mathsf{SH}}$ sends randomness $r_i^{\mathsf{Setup}}, r_i^{\mathsf{Enc}}, r_i^{\mathsf{SH}}, r_i^{\mathsf{KeyGen}}$.

In the interaction with the challenger, $\mathcal{A}_{\mathsf{SH}}$ receives messages $(\mathsf{msg}_1, \mathsf{msg}_2)$ for rounds 1 and 2 of protocol π^{SH} for each honest party P_j from the challenger $\mathcal{C}_{\mathsf{SH}}$. $\mathcal{A}_{\mathsf{SH}}$ sets these as the messages $(\mathsf{msg}_{1,j}, \mathsf{msg}_{2,j})$ in its interaction with $\mathcal{A}$. The rest of the experiment with $\mathcal{A}$ (i.e generating SK_j for each honest P_j) is performed exactly as in Hyb_0.

Notice that when the challenger $\mathcal{C}_{\mathsf{SH}}$ sends honestly generates messages for the honest parties (real world), the experiment between $\mathcal{A}_{\mathsf{SH}}$ and $\mathcal{A}$ corresponds exactly to Hyb_0 and when the challenger $\mathcal{C}_{\mathsf{SH}}$ sends simulated messages for the honest parties (ideal world), the experiment corresponds exactly to Hyb_1. Thus, if $\mathcal{A}$ can distinguish between the two hybrids with non-negligible probability, $\mathcal{A}_{\mathsf{SH}}$ can use the same guess to break the security of the MPC protocol π^{SH} with non-negligible probability which is a contradiction. □

Lemma 9. *Assuming* $\mathsf{DFEComb}$ *is a succinct single-key simulation secure decomposable FE combiner scheme, Hyb_1 is computationally indistinguishable from Hyb_2.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that can distinguish between the two hybrids with non-negligible probability. We will use $\mathcal{A}$ to construct an adversary $\mathcal{A}_{\mathsf{DFEComb}}$ that breaks the security of the succinct simulation secure decomposable FE combiner $\mathsf{DFEComb}$ which is a contradiction.

The adversary $\mathcal{A}_{\mathsf{DFEComb}}$ interacts with the adversary $\mathcal{A}$ as in Hyb_1. For each $i \in \mathsf{Corr}$, $\mathcal{A}_{\mathsf{DFEComb}}$ receives $(m_i, r_i^{\mathsf{Setup}}, r_i^{\mathsf{Enc}}, r_i^{\mathsf{SH}}, r_i^{\mathsf{KeyGen}})$ from $\mathcal{A}$. Then, $\mathcal{A}_{\mathsf{DFEComb}}$ computes $(C_1, \ldots, C_n) \leftarrow \mathsf{DFEComb.Partition}(1^\lambda, C)$.

For each honest party P_j, $\mathcal{A}_{\mathsf{DFEComb}}$ picks $(m_j, r_j^{\mathsf{Setup}}, r_j^{\mathsf{Enc}}, r_j^{\mathsf{KeyGen}})$ randomly

and computes $\mathsf{MSK}_j \leftarrow \mathsf{FE}_j.\mathsf{Setup}(1^\lambda)$, $\mathsf{SK}_j \leftarrow \mathsf{FE}_j.\mathsf{KeyGen}(\mathsf{MSK}_j, C_j)$ using randomness r_j^{Setup} and r_j^{KeyGen} respectively. Then, $\mathcal{A}_{\mathsf{DFEComb}}$ picks an honest party P_j at random. Let I denote the set of indices of all parties except P_{j^*}. interacts with the challenger $\mathcal{C}_{\mathsf{DFEComb}}$ and sends the following tuple: $(\{m_i, \mathsf{MSK}_i, \mathsf{SK}_i, r_i^{\mathsf{Enc}}, C, C(m_1, \ldots, m_n)\}_{i\in[n]}, I)$. $\mathcal{A}_{\mathsf{DFEComb}}$ receives back a ciphertext CT from $\mathcal{C}_{\mathsf{DFEComb}}$ that is either honestly generated or simulated. It sets this as the ciphertext CT^* in its interaction with $\mathcal{A}$. The rest of the experiment with $\mathcal{A}$ is performed exactly as in Hyb_1.

Notice that when the challenger $\mathcal{C}_{\mathsf{DFEComb}}$ sends an honestly generated ciphertext, the experiment between $\mathcal{A}_{\mathsf{DFEComb}}$ and $\mathcal{A}$ corresponds exactly to Hyb_1 and when the challenger $\mathcal{C}_{\mathsf{SH}}$ sends a simulated ciphertext, the experiment corresponds exactly to Hyb_2. Thus, if $\mathcal{A}$ can distinguish between the two hybrids with non-negligible probability, $\mathcal{A}_{\mathsf{DFEComb}}$ can use the same guess to break the security of the scheme $\mathsf{DFEComb}$ with non-negligible probability which is a contradiction. □

6 Construction of an FE Combiner from Weaker Assumptions

In this section, we employ a tool extensively used in the secure multi-party computation literature, namely, randomized encodings to construct an FE combiner. Roughly speaking, a randomized encoding is a mechanism to "efficiently" encode a function f and an input x such that the encoding reveals $f(x)$ and nothing more. A randomized encoding scheme is said to be low degree if the encoding algorithm can be represented as a low degree polynomial. Low degree randomized encodings have been used to achieve constant-round secure multi-party computation [BMR90]. We show how to use this tool to obtain functional encryption combiners. The underlying assumption used to instantiate the low degree randomized encoding is the existence of a PRG in NC^1. Formally, we show the following theorem.

Theorem 8. *Assuming the existence of a PRG in NC^1, there exists an FE combiner for polynomial-sized circuits.*

6.1 Degree-d Randomizing Polynomials

At the heart of our FE combiner construction are constant degree randomizing polynomials schemes. Such randomizing polynomials schemes can be defined as follows.

Definition 11. *A degree-d randomizing polynomials scheme, $\mathsf{RP} = (\mathsf{Encode}, \mathsf{Decode})$, for a class of circuits $\mathcal{C}$ over some field $\mathbb{F}$ consists of the following polynomial time algorithms:*

- *Encoding, $\mathsf{Encode}(1^\lambda, C, x; r)$: On input the security parameter λ, a circuit C, an input x and some randomness r, Encode outputs an encoding of $C \in \mathcal{C}$ and x, denoted $\widehat{C, x}$. We require that $\widehat{C, x} = (p_1(x, r), ..., p_\ell(x, r))$, where each p_i is a degree d homogenous polynomial over $\mathbb{F}$ dependent only on C.*
- *Decoding, $\mathsf{Decode}(\widehat{C, x})$: On input an encoding of C and x, $\widehat{C, x}$, Decode outputs the decoded value α.*

A randomizing polynomials scheme, RP, is required to satisfy the following properties:

- **Correctness**: *For every $\lambda \in \mathbb{N}$, circuit $C \in \mathcal{C}$ and input x, $C(x) = \mathsf{Decode}(\mathsf{Encode}(1^\lambda, C, x))$.*
- **Security**: *For every PPT adversary $\mathcal{A}$, large enough $\lambda \in \mathbb{N}$, circuit $C \in \mathcal{C}$ and input x, there exists a simulator SimRP such that:*

$$\{\mathsf{Encode}(1^\lambda, C, x)\} \approx \{\mathsf{SimRP}(1^\lambda, |C|, C(x))\}$$

Theorem 9 ([AIK05])**.** *Assuming the existence of a PRG in NC^1, there exist constant degree randomizing polynomials schemes for polynomial-sized circuits.*

6.2 d-Nested FE

Another tool used in our construction is d-nested FE. d-nested FE is a new FE candidate that can be created easily from d FE candidates by simply encrypting in sequence using the d FE candidates. Intuitively, this new FE candidate will be secure as long as one of the d candidates is secure since an adversary should be unable to break the encryption of the secure candidate. d-nested FE can be viewed as an FE combiner that can only handle a constant number of FE candidates since the runtime of its algorithms may depend exponentially on d. We present the definition of d-nested FE below.

Definition 12 (d-nested FE)**.** *Let $\mathsf{FE}_1, ..., \mathsf{FE}_d$ be d correct FE candidates. We define another FE candidate FE_S, where $S = [1, .., d]$, as follows:*

- $\mathsf{Setup}(1^\lambda)$: *It computes* $\mathsf{MSK}_i \leftarrow \mathsf{FE}_i.\mathsf{Setup}(1^\lambda)$ *for all* $i \in [n]$ *and outputs* $\mathsf{MSK} = \{\mathsf{MSK}_i\}_{i\in[n]}$ *as the master secret key for* FE_S.

- $\mathsf{Enc}(\mathsf{MSK} = \{\mathsf{MSK}_1, .., \mathsf{MSK}_d\}, m)$: *It first computes* $\mathsf{CT}_1 = \mathsf{FE}_1.\mathsf{Enc}(\mathsf{MSK}_1, m)$ *and then recursively computes* $\mathsf{CT}_{i+1} = \mathsf{FE}_{i+1}.\mathsf{Enc}(\mathsf{MSK}_{i+1}, \mathsf{CT}_i)$ *for* $i \in [d-1]$. *It outputs* CT_d *as the ciphertext.*

- $\mathsf{KeyGen}(\mathsf{MSK} = \{\mathsf{MSK}_1, .., \mathsf{MSK}_d\}, C)$: *On input the master secret key* MSK *and a circuit* $C \in \mathcal{C}$, *it first computes* $\mathsf{SK}_1 = \mathsf{FE}_1.\mathsf{KeyGen}(\mathsf{MSK}_1, C)$ *and sets* $G_1 = \mathsf{FE}_1.\mathsf{Dec}(\mathsf{SK}_1, \cdot)$ *as a circuit. Then it recursively computes* $\mathsf{SK}_{i+1} = \mathsf{FE}_{i+1}.\mathsf{KeyGen}(\mathsf{MSK}_{i+1}, G_i)$ *for* $i \in [d-1]$ *where circuit* $G_i = \mathsf{FE}_i.\mathsf{Dec}(\mathsf{SK}_i, \cdot)$. *It outputs* SK_d *as the function secret key for the circuit* C.

- $\mathsf{Dec}(\mathsf{SK}_d, \mathsf{CT}_d)$: *It outputs* $\mathsf{FE}_d.\mathsf{Dec}(\mathsf{SK}_d, \mathsf{CT}_d)$.

Theorem 10. *For constant* d, FE_S, *defined by Definition 12, is an FE candidate. Moreover, if at least one of the* d *FE candidates is secure, then* FE_S *is also secure (it is an FE scheme).*

Proof. We observe that if d is a constant, the scheme described above is efficient if the underlying candidates are efficient. Correctness of FE_S follows immediately from the construction and the correctness of the underlying FE candidates.

Suppose that one of the FE candidates, say FE_j, is secure. Suppose that FE_S is not secure; that is, there exists an adversary $\mathcal{A}$ that can break the security of FE_S. Then, consider the following adversary $\mathcal{A}'$ that breaks the security of FE_j. $\mathcal{A}'$ runs $\mathcal{A}$ and simulates the challenger. $\mathcal{A}'$ first runs $\mathsf{FE}_i.\mathsf{Setup}(1^\lambda)$ for all $i \in [n] \setminus \{j\}$ to obtain MSK_i's. When $\mathcal{A}$ submits a challenge message pair (m_0, m_1), $\mathcal{A}'$ computes $\mathsf{CT}_{j-1,0}$ and $\mathsf{CT}_{j-1,1}$ by encrypting m_0 and m_1, respectively by repeated encryption under $\mathsf{FE}_1, \ldots, \mathsf{FE}_{j-1}$. It then submits these two ciphertexts as its messages to its challenger and receives a ciphertext, CT_j. It then encrypts this ciphertext using $\mathsf{FE}_{j+1}, \ldots, \mathsf{FE}_d$ to obtain CT_d, which it gives to $\mathcal{A}$ as the ciphertext. When $\mathcal{A}$ asks for a function key for a circuit C, $\mathcal{A}'$ runs $\mathsf{FE}_S.\mathsf{KeyGen}$ on C until it needs to run the keygen algorithm for FE_j. Here, it queries its challenger for the appropriate function key and then uses this response to continue generating SK_C. It then gives the resulting function key SK_C to $\mathcal{A}$. When $\mathcal{A}$ terminates, $\mathcal{A}'$ outputs the same response. Note that $\mathcal{A}'$ simulates the security game for $\mathcal{A}$ perfectly and therefore wins whenever $\mathcal{A}$ wins, contradicting the security of FE_j. Therefore, it follows that FE_S is a secure FE scheme. □

6.3 Construction

We are now ready to prove Theorem 8. At a high level, the construction works as follows. For a circuit C, we use a constant degree randomizing polynomials scheme to obtain polynomials p_α corresponding to C. Since $C(x)$ can be obtained from the evaluations of these polynomials on (x, r) for randomness r, it suffices to be able to generate function keys for degree-d homogeneous polynomials. Additionally, since only one of the FE candidates is guaranteed to be secure, we need a means of ensuring that no information is leaked even if all but one FE candidate is broken. This is accomplished by bitwise secret sharing the input x, with one share per FE candidate. We then make the observation that degree-d homogeneous polynomials in the bits of x remain degree d homogeneous polynomials when expressed as polynomials in the *shares* of the bits of x. Furthermore, we observe that the number of subsets $S \subseteq [n]$ of size at most d is polynomial in n. We make the ciphertext for x consist of a set of ciphertexts, one for each subset S, and generate them using the d-nested FE candidates, FE_S. Each of these ciphertexts contains the shares of x corresponding to the FE candidates used by FE_S. Since every monomial in the polynomials has degree d, we can map each monomial to a corresponding set S such that

it is possible to evaluate the monomial using the contents of the ciphertext corresponding to S. By masking the monomial evaluations with secret shares of 0, as described in the technical overview, we are able to ensure that only the value of the polynomial is learned.

We now formally describe the construction. First, we provide some notation that will be used throughout the construction.

Notation:

- Let PRF be a pseudorandom function that outputs in the range $\{0,1\}^\lambda$.
- Let E be any λ-bit CPA-secure secret-key encryption scheme with message space $\{0,1\}$.
- Let $\ell_x = \ell_x(\lambda)$ denote the length of the messages and let $\ell_{\mathsf{E}} = \ell_{\mathsf{E}}(\lambda)$ denote the length of the encryption key for the scheme E.
- Let RP denote a degree d randomizing polynomials scheme over $\mathbb{F}_2$ for the circuit class $\mathcal{C}$, the class of all polynomial-sized boolean circuits with input space $\{0,1\}^{\ell_x}$.
- Let $\mathsf{FE}_1, \ldots, \mathsf{FE}_n$ denote n FE candidates. In the following construction, we assume that the descriptions $\{\mathsf{FE}_i\}_{i\in[n]}$ are implicit in all the algorithms of FEComb.

Construction:

- $\mathsf{FEComb.Setup}(1^\lambda)$: On input the security parameter, it runs $\mathsf{MSK}_i \leftarrow \mathsf{FE}_i.\mathsf{Setup}(1^\lambda)$ for $i \in [n]$ and $\mathsf{E.SK} \leftarrow \mathsf{E.Setup}(1^\lambda)$. It outputs $\mathsf{MSK} = (\mathsf{MSK}_1, .., \mathsf{MSK}_n, \mathsf{E.SK})$.

- $\mathsf{FEComb.Enc}(\mathsf{MSK}, x \in \{0,1\}^{\ell_x})$: It executes the following steps.
  - First, randomly secret share x amongst n shares using bitwise additive secret sharing. That is, for each x_i for $i \in [\ell_x]$, $x_i = \Sigma_{j\in[n]} x_{i,j}$ where $x_{i,j}$ is j^{th} share of x_i.
  - Then, sample n distinct PRF keys K_i for $i \in [n]$ uniformly at random.
  - For each subset $S \subseteq [n]$ of size at most d, compute

$$\mathsf{CT}_S = \mathsf{FE}_S.\mathsf{Enc}\left(\{\mathsf{MSK}_j\}_{j\in S}, (\{x_{i,j}, K_j\}_{i\in[\ell_x], j\in S}, 0^{\ell_{\mathsf{E}}}, 0)\right).$$

  - Output $\mathsf{CT} = \{\mathsf{CT}_S\}_{S\subseteq[n], 1\le|S|\le d}$.

- $\mathsf{FEComb.KeyGen}(\mathsf{MSK}, C)$: Generate $tag_C \leftarrow \{0,1\}^\lambda$ uniformly at random. Let $p_1, .., p_N$ denote the randomizing polynomials corresponding to the circuit C determined by the encoding function of RP. For each polynomial p_α, do the following:
  - Let $p_\alpha = p_\alpha(x_1, .., x_{\ell_x}, r_1, .., r_{\ell_r})$ where ℓ_r is the length of the randomness taken by $\mathsf{RP.Encode}$. Define new variables $x_i = \Sigma_{j\in[n]} x_{i,j}$ for $i \in [\ell_x]$ and $r_i = \Sigma_{j\in[n]} r_{i,j}$ for $i \in [\ell_r]$. Let p'_α be the degree d polynomial obtained by computing p_α as a polynomial over the variables $x_{i,j}$ and $r_{i,j}$.
  - For each polynomial p'_α do the following.
    - Generate $tag_{p'_\alpha} \leftarrow \{0,1\}^\lambda$ uniformly at random (and distinct for each polynomial).
    - Let M_α be the number of monomials occurring in p'_α. For every $j \in [n]$, denote by $M_{\alpha,j}$ the number of monomials depending on variables of the form $x_{i',j}$ or $r_{i',j}$ for some i'. Viewing the $x_{i',j}$'s and $r_{i',j}$'s as the jth party's shares of the x_i's and r_i's, $M_{\alpha,j}$ is the number of monomials in p'_α that contain a share corresponding to the jth party.

* Let there be a total ordering on the monomials of p'_α. For all monomials $m_{\alpha,k}$ for $k \in [M_\alpha]$ of p'_α, let $S_{m_{\alpha,k}}$ denote the set of size less than or equal to d consisting of the parties whose shares of variables lie in $m_{\alpha,k}$. That is, if $m_{\alpha,k} = x_{i_1,j_1}x_{i_2,j_2}\ldots r_{i_d,j_d}$, then $S_{m_{\alpha,k}} = \{j_1, j_2, \ldots, j_d\}$.
* For this ordering, let $V(p'_\alpha, j, m_{\alpha,k})$ denote the number of monomials (less than or equal to $m_{\alpha,k}$) that have a variable of the form $x_{i',j}$ or $r_{i',j}$ (a share corresponding to the jth party).
* For every monomial $m_{\alpha,k}$ occuring inside p'_α, first generate a ciphertext $c_{m_{\alpha,k}}$ by running

$$\mathsf{E.Enc}(\mathsf{E.SK}, \beta_{m_{\alpha,k}}),$$

where $\beta_{m_{\alpha,k}}$ is a bit chosen uniformly at random. Then, compute

$$\mathsf{SK}_{p'_\alpha,m_{\alpha,k}} \leftarrow \mathsf{FE}_{S_{m_{\alpha,k}}}.\mathsf{KeyGen}(\{\mathsf{MSK}_j\}_{j\in S_{m_{\alpha,k}}}, H_{p'_\alpha,m_{\alpha,k}}),$$

where circuit $H_{p'_\alpha,m_{\alpha,k}}$ is described in Figure 3. Here FE_S for any subset $S \subseteq [d]$, is the $|S|-$nested FE scheme.

$H_{p'_\alpha,m_{\alpha,k}}$

Input: Shares of input $\{x_{i,j}\}_{i\in[\ell_x],j\in S_{m_{\alpha,k}}}$, PRF keys $\{K_j\}_{j\in S_{m_{\alpha,k}}}$, a string $s \in \{0,1\}^{\ell_\mathsf{E}}$, and a bit b

Hardwired: Polynomial p'_α, monomial $m_{\alpha,k}$, ciphertext $c_{m_{\alpha,k}}$, tags $tag_C \in \{0,1\}^\lambda$ and $tag_{p'_\alpha} \in \{0,1\}^\lambda$

- If $b \neq 0$, output $\mathsf{E.Dec}(s, c_{m_{\alpha,k}})$. Otherwise, do the following.
- Set $r_{i',j} = \mathsf{PRF}_1(K_j, i'||tag_C)$ for $i' \in [\ell_r]$ and $j \in S_{m_{\alpha,k}}$. Here PRF_1 denotes the output of the PRF truncated to one bit.
- Compute monomial $m_{\alpha,k}$ by substituting values of $\{x_{i,j}\}_{i\in[\ell_x]}$ and $\{r_{i',j}\}_{i'\in[\ell_r]}$. Let this value be stored as $u_{\alpha,k}$.
- For all $j \in S_{m_{\alpha,k}}$, compute an additive secret sharing of 0 consisting of $M_{\alpha,j}$ shares using the randomness $\mathsf{PRF}(K_j, tag_{p'_\alpha})$. Denote these shares as $(\mathsf{Share}_{\alpha,j,1}, .., \mathsf{Share}_{\alpha,j,M_{\alpha,j}})$.
- Output $u_{\alpha,k} + \Sigma_{j\in S_{m_{\alpha,k}}} \mathsf{Share}_{\alpha,j,V(p'_\alpha,j,m_{\alpha,k})}$

Figure 3: Description of the Monomial Evaluation Circuit.

– Output $\mathsf{SK}_C = (C, \{\mathsf{SK}_{p'_\alpha,m_{\alpha,k}}\})$, where the set is the set of function keys for each monomial $m_{\alpha,k}$ of each polynomial p'_α.

- $\mathsf{FEComb.Dec}(\mathsf{SK}_C, \mathsf{CT})$: Using C, derive p'_α for all $\alpha \in [N]$ and each of its monomials $m_{\alpha,k}$ for $k \in [M_\alpha]$. For all function keys $\mathsf{SK}_{p'_\alpha,m_{\alpha,k}} \in \mathsf{SK}_C$, compute

$$v_{p'_\alpha,m_{\alpha,k}} = \mathsf{FE}_{S_{m_{\alpha,k}}}.\mathsf{Dec}(\mathsf{SK}_{p'_\alpha,m_{\alpha,k}}, \mathsf{CT}).$$

Compute

$$v_{p'_\alpha} = \sum_{k\in[M_\alpha]} v_{p'_\alpha,m_{\alpha,k}}$$

for all $\alpha \in [N]$. Run $\mathsf{RP.Decode}((v_{p'_1}, \ldots, v_{p'_N}))$ and output the result.

Correctness: To see that the above construction satisfies correctness, let $\{\mathsf{FE}_i\}_{i\in[n]}$ be n FE candidates, let $C : \{0,1\}^{\ell_x(\lambda)} \to \mathcal{Y}_\lambda \in \mathcal{C}_\lambda$ be any circuit and let $x \in \{0,1\}^{\ell_x(\lambda)}$ be any message.

Suppose we run $\mathsf{FEComb.Setup}(1^\lambda)$ to obtain the master secret key MSK, then encrypt x by running $\mathsf{FEComb.Enc}(\mathsf{MSK}, x)$ to obtain CT, and then generate a function key SK_C by running $\mathsf{FEComb.Keygen}(\mathsf{MSK}, C)$. Then, if we run $\mathsf{FEComb.Dec}(\mathsf{SK}_C, \mathsf{CT})$, we note the following. By the correctness of the underlying FE candidates, the values $v_{p'_\alpha, m_{\alpha,k}}$ will be

$$u_{\alpha,k} + \Sigma_{j\in S_{m_{\alpha,k}}} \mathsf{Share}_{\alpha,j,V(p'_\alpha,j,m_{\alpha,k})}.$$

Then,

$$v_{p'_\alpha} = \sum_{k\in[M_\alpha]} v_{p'_\alpha, m_{\alpha,k}} = \sum_{k\in[M_\alpha]} u_{\alpha,k}$$

is the value of the polynomial p_α on input x with randomness

$$r_i = \sum_{j\in[n]} \mathsf{PRF}_1(K_j, i||tag_C)$$

for $i \in [\ell_r]$. Since the p_α's are the polynomials corresponding to C, by the correctness of RP, it follows that $\mathsf{RP.Decode}((p_1(x;r), \ldots, p_N(x;r))) = C(x)$ as desired.

Polynomial Slowdown: All the algorithms of FEComb run in time $\mathsf{poly}(\lambda, n)$. This follows immediately from the efficiency of the FE candidates, RP, and E and the fact that all subsets $S \subseteq [n]$ of size at most d is $\mathsf{poly}(n)$ since d is a constant.

6.4 Security Proof

For simplicity, we will consider the case where the adversary submits a single message pair and only queries a single function key. This proof naturally extends to multiple message pairs and multiple function keys via standard hybrid techniques (repeat the series of hybrids for each function key sequentially to argue indistinguishability for a polynomial number of function keys and then flip each message pair one at a time to argue indistinguishability for a polynomial number of message pairs). We show that any PPT adversary $\mathcal{A}$ will only succeed in the FE selective security game (Definition 15) with negligible probability. We will show this via a sequence of hybrids. Assume that the γth FE candidate, FE_γ is secure.

Hyb_0**:** This hybrid corresponds to the real security game where the challenger sets the bit b to 0.

Hyb_1**:** This hybrid is the same as Hyb_0 except that when generating $c_{m_{\alpha,k}}$, the challenger sets $\beta_{m_{\alpha,k}}$ to be

$$u_{\alpha,k} + \Sigma_{j\in S_{m_{\alpha,k}}} \mathsf{Share}_{\alpha,j,V(p'_\alpha,j,m_{\alpha,k})},$$

the value obtained by running $H_{p'_\alpha, m_{\alpha,k}}$ on $\{x_{i,j}\}_{i\in[\ell_x], j\in S_{m_{\alpha,k}}}$, $\{K_j\}_{j\in S_{m_{\alpha,k}}}$, $\mathsf{E.SK}$, and 0. Note that the challenger knows these values because $\mathcal{A}$ must submit x and the challenger generates the encryption of x prior to generating the function key SK_C.

Hyb_2**:** This hybrid is the same as Hyb_1 except that when encrypting x, whenever the challenger generates CT_S for a subset S that contains γ, the challenger does so by setting

$$\mathsf{CT}_S = \mathsf{FE}_S.\mathsf{Enc}\left(\{\mathsf{MSK}_j\}_{j\in S}, (\{x_{i,j}, K_j\}_{i\in[\ell_x], j\in S}, \mathsf{E.SK}, 1)\right).$$

That is, the $0^{\ell_{\mathsf{E}}}$ term is replaced with $\mathsf{E.SK}$, the secret key for E, and the last bit is switched to 1 when generating ciphertexts for subsets S that contain γ.

Hyb_3**:** This hybrid is the same as Hyb_2 except that when encrypting x, whenever the challenger generates CT_S for a subset S that contains γ, the challenger replaces K_γ, the PRF key for the γth party, with the all 0 string.

Hyb_4**:** This hybrid is the same as Hyb_3 except that when setting $\beta_{m_{\alpha,k}}$'s, the additive secret sharing of 0 consisting of $M_{\alpha,\gamma}$ shares is computed using uniformly sampled randomness rather than $\mathsf{PRF}(K_\gamma, tag_{p'_\alpha})$ and the randomness shares $r_{i',\gamma}$ corresponding to the γth party are generated uniformly at random, as opposed to using $\mathsf{PRF}_1(K_\gamma, i'||tag_C)$. That is, every evaluation output of the PRF that used K_γ as the PRF key is replaced by a uniformly random value.

Hyb_5**:** This hybrid is the same as Hyb_4 except that when encrypting the $x_{i,\gamma}$'s, (the shares of x corresponding to the γth party), the challenger instead encrypts new values $x'_{i,\gamma}$, sampled uniformly at random.

Hyb_6**:** This hybrid is the same as Hyb_5 except the challenger now uses the randomizing polynomials simulator, SimRP, to generate the values of the polynomials. That is, when generating a function key for a circuit C, the challenger first runs

$$\mathsf{SimRP}(1^\lambda, |C|, C(x))$$

to obtain $v_{p'_\alpha}$, the values for all the randomizing polynomials. The challenger then proceeds with the key generation as before, except when computing the $\beta_{m_{\alpha,k}}$ value for a monomial $m_{\alpha,k}$ where $\gamma \in S_{m_{\alpha,k}}$. Let $\mathcal{M}_\alpha$ denote the set containing all monomials of p'_α. Let $\mathcal{M}_{\alpha,\gamma}$ denote the set of all monomials of p'_α that contain a variable of the form $x_{i',\gamma}$ or $r_{i',\gamma}$ for some i'. In this hybrid, the challenger first sets $\beta_{m_{\alpha,k}}$ for all $m_{\alpha,k} \notin \mathcal{M}_{\alpha,\gamma}$ as before. Then, the challenger computes

$$v'_{p'_\alpha} = v_{p'_\alpha} - \sum_{m_{\alpha,k} \in \mathcal{M}_\alpha \setminus \mathcal{M}_{\alpha,\gamma}} \beta_{m_{\alpha,k}}.$$

The challenger then secret shares the value $v'_{p'_\alpha}$ uniformly at random amongst $|\mathcal{M}_{\alpha,\gamma}|$ shares and sets each of the $\beta_{m_{\alpha,k}}$ values for the monomials $m_{\alpha,k} \in \mathcal{M}_{\alpha,\gamma}$ to be such that

$$\sum_{m_{\alpha,k} \in \mathcal{M}_{\alpha,\gamma}} \beta_{m_{\alpha,k}} = v'_{p'_\alpha}.$$

Lemma 10. *If* E *is a* λ*-bit CPA-secure secret-key encryption scheme, then* Hyb_0 *and* Hyb_1 *are computationally indistinguishable.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that can distinguish between these two hybrids. Then, we construct an adversary $\mathcal{A}'$ that can break the security game of E. Since $\mathcal{A}$ can distinguish between Hyb_0 and Hyb_1 and since there are at most a polynomial number of monomials $m_{\alpha,k}$, we can construct a sequence of hybrids that correspond to switching the values $\beta_{m_{\alpha,k}}$ one at a time. It follows that $\mathcal{A}$ must be able to distinguish between two of these neighboring hybrids. WLOG, say $\mathcal{A}$ distinguishes between the sequential hybrids where the $\beta_{m_{\alpha',k'}}$ value is changed. Then, $\mathcal{A}'$ runs as follows. It runs $\mathcal{A}$ and plays the role of the challenger. Whenever, it would have to generate a ciphertext by running $\mathsf{E.Enc}$, it instead queries its challenger on the message to receive the ciphertext. When it wants to encrypt the value corresponding to $m_{\alpha',k'}$, it submits the message pair $(\beta^0_{m_{\alpha',k'}}, \beta^1_{m_{\alpha',k'}})$ to its challenger where the first component is the unchanged (randomly sampled) value and the second component is the changed (monomial evaluation) value corresponding to Hyb_0 and Hyb_1, respectively. It then uses the

ciphertext it receives from its challenger. When $\mathcal{A}$ outputs, $\mathcal{A}'$ outputs the same response. Note that $\mathcal{A}'$ simulates $\mathcal{A}$ on these two neighboring hybrids perfectly and so $\mathcal{A}'$ will distinguish with nonnegligible advantage, contradicting the security of E. Therefore, Hyb_0 and Hyb_1 must be indistinguishable. □

Lemma 11. *If* FE_γ *is a selectively-secure FE scheme, then* Hyb_1 *and* Hyb_2 *are computationally indistinguishable.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that can distinguish between these two hybrids. Then, we construct an adversary $\mathcal{A}'$ that can break the security game of FE_γ. $\mathcal{A}'$ runs $\mathcal{A}$ and simulates the role of the challenger. Whenever, it needs to encrypt using $\mathsf{FE}_\gamma.\mathsf{Enc}$, it computes the messages m_1 and m_2 that it would want to encrypt were it simulating Hyb_1 or Hyb_2, respectively. It then submits (m_1, m_2) as a message pair to its challenger and receives a ciphertext, CT, which it uses to continue its simulation. Note that $\mathcal{A}'$ can submit all its message queries prior to submitting any function queries. Whenever it would need to run $\mathsf{FE}_\gamma.\mathsf{KeyGen}$ on a circuit C, it submits C as a function query and uses the response of its challenger to continue the simulation. When $\mathcal{A}$ terminates, $\mathcal{A}'$ outputs the same value as $\mathcal{A}$. Note that $\mathcal{A}'$ simulates both Hyb_1 and Hyb_2 exactly and is an admissible adversary since all computable function evaluations are identical across hybrids, so $\mathcal{A}'$ wins if $\mathcal{A}$ wins, contradicting the security of FE_γ. □

Lemma 12. *If* FE_γ *is a selectively-secure FE scheme, then* Hyb_2 *and* Hyb_3 *are computationally indistinguishable.*

Proof. This follows from an analogous proof as that of Lemma 11. □

Lemma 13. *If* PRF *is a pseudorandom function, then* Hyb_3 *and* Hyb_4 *are computationally indistinguishable.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that can distinguish between these two hybrids. Then, we construct an adversary $\mathcal{A}'$ that can break the pseudorandomness property of PRF. $\mathcal{A}'$ simply runs $\mathcal{A}$ and simulates the challenger. Whenever $\mathcal{A}'$ would run $\mathsf{PRF}(K_\gamma, \cdot)$ on some input t in Hyb_3 or generate a uniformly random value in Hyb_4, $\mathcal{A}'$ instead queries its oracle on t and uses this value as its output. If $\mathcal{A}$ thinks it is seeing Hyb_3, $\mathcal{A}'$ outputs that its oracle is the PRF and if $\mathcal{A}$ thinks it is seeing Hyb_4, $\mathcal{A}'$ outputs that its oracle is a truly random function. Note that $\mathcal{A}'$ simulates these hybrids exactly, so $\mathcal{A}'$ will win with nonnegligible advantage, contradicting the pseudorandomness of PRF. □

Lemma 14. *If* FE_γ *is a selectively-secure FE scheme, then* Hyb_4 *and* Hyb_5 *are computationally indistinguishable.*

Proof. This follows from an analogous proof as that of Lemma 11. □

Lemma 15. *If* RP *is a secure randomizing polynomials scheme, then* Hyb_5 *and* Hyb_6 *are computationally indistinguishable.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that can distinguish between these two hybrids. Then, we construct an adversary $\mathcal{A}'$ that can break the security of RP. $\mathcal{A}'$ runs $\mathcal{A}$ and simulates the role of the challenger. When it needs to generate a function key for a circuit C, it queries its challenger of circuit C and input x (specified by $\mathcal{A}$). It then receives values $v_{p'_\alpha}$ for all polynomials. $\mathcal{A}'$ sets the $\beta_{m_{\alpha,k}}$ values as follows. Let $\mathcal{M}_\alpha$ denote the set containing all monomials of p'_α. Let $\mathcal{M}_{\alpha,\gamma}$ denote the set of all monomials of p'_α that contain a variable of the form $x_{i',\gamma}$ or $r_{i',\gamma}$ for some i'. $\mathcal{A}'$ first sets $\beta_{m_{\alpha,k}}$ for all $m_{\alpha,k} \notin \mathcal{M}_{\alpha,\gamma}$ as specified by both Hyb_5 and Hyb_6. Then, $\mathcal{A}'$ computes

$$v'_{p'_\alpha} = v_{p'_\alpha} - \sum_{m_{\alpha,k} \in \mathcal{M}_\alpha \setminus \mathcal{M}_{\alpha,\gamma}} \beta_{m_{\alpha,k}}.$$

$\mathcal{A}'$ then secret shares the value $v'_{p'_\alpha}$ uniformly at random amongst $|\mathcal{M}_{\alpha,\gamma}|$ shares and sets each of the $\beta_{m_{\alpha,k}}$ values for the monomials $m_{\alpha,k} \in \mathcal{M}_{\alpha,\gamma}$ to be such that

$$\sum_{m_{\alpha,k} \in \mathcal{M}_{\alpha,\gamma}} \beta_{m_{\alpha,k}} = v'_{p'_\alpha}.$$

When $\mathcal{A}$ terminates, if $\mathcal{A}$ guesses that the game is Hyb_5, then $\mathcal{A}'$ outputs that it is using the real encoding function. If $\mathcal{A}$ guesses that the game is Hyb_6, then $\mathcal{A}'$ outputs that its challenger is the simulator. Note that $\mathcal{A}'$ simulates these hybrids exactly. This follows from the fact that all the $\beta_{m_{\alpha,k}}$ values for $m_{\alpha,k} \in \mathcal{M}_{\alpha,\gamma}$ are masked by uniformly random secret shares of 0. Therefore, the values of these $\beta_{m_{\alpha,k}}$'s look like a uniformly random secret share of $v'_{p'_\alpha}$, their intended sum and Hyb_5 is simulated exactly. If $\mathcal{A}'$'s challenger is using the simulated encoding, then $\mathcal{A}'$ simulates Hyb_6 exactly. So, it follows that $\mathcal{A}'$ could break the security of RP, a contradiction. □

So, it follows that Hyb_0 and Hyb_6 are computationally indistinguishable. Moreover, Hyb_6 is independent of x and depends only on $C(x)$. Therefore, by an analogous sequence of hybrids, this time with the challenge bit $b = 1$, it follows that the real game with $b = 1$ is also indistinguishable from Hyb_6. So, it follows that the two real games are computationally indistinguishable, completing the proof. □

7 From MPC to FE Combiners

In this section, we show how to build an FE combiner from any semi-honest MPC protocol π that satisfies a property called delayed function-dependence. Informally, we say that an MPC protocol is *delayed function-dependent* if the messages of all the rounds except the last one can be generated independently of the function being evaluated. Formally, let π denote a k round semi-honest MPC protocol for computing a function f between n parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$ using inputs $x_1, \ldots, x_n$, respectively. For each $i \in [k]$, let $\pi^{\mathsf{SH}}.\mathsf{Round}_i$ denote the algorithm used by each party to compute its message for round i. Let $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ denote the simulator for π where Sim_1 is used to generate the first $(k-1)$ rounds of the adversary's view and Sim_2 generates the last round's view.

Definition 13 (Delayed Function-Dependent)**.** *Protocol π is said to be* delayed function-dependent *if, for every function f and all inputs $(x_1, \ldots, x_n)$:*

- ***Correctness:*** *For each party P_j, for each $i \in [k-1]$, algorithm $\pi^{\mathsf{SH}}.\mathsf{Round}_i$ only takes as input $(x_j, \tau, |f|)$ where τ is the transcript of π up to round $(i-1)$. In particular, it does not take as input the function f being computed.*
- ***Security:*** *Let $\mathcal{A}$ denote the set of corrupted parties. Algorithm Sim_1 only takes as input $(|f|, \{x_i\}_{\mathsf{P}_i \in \mathcal{A}})$. In particular, it does not take as input the function f or the output $f(x_1, \ldots, x_n)$.*

We observe that several MPC protocols in literature [MW16, PS16, BP16] are, in fact, delayed function-dependent.

Using any delayed function-dependent MPC protocol, we are able to construct an FE combiner. Formally, we show the following theorem:

Theorem 11. *Given any delayed function-dependent semi-honest MPC protocol π secure against all but one corruption, there exists a bounded-collusion secure FE combiner.*

Furthermore, we can construct an unbounded-collusion secure FE combiner if the delayed function-dependent MPC protocol π additionally satisfies the following *reusability* property.

Consider n parties with inputs $x_1, \ldots, x_n$. Run the first $(k-1)$ rounds of the protocol π to generate a transcript τ. Informally, reusability requires that the parties should be able to reuse this same transcript τ to securely evaluate an unbounded number of functions $f_1, \ldots, f_\ell$. That is, for each function f_i, they can just run the last round of π using transcript τ to compute the output and the transcript τ along with the set of outputs should not reveal any party's input. (beyond what can be learned from the given information). Formally, consider a k round delayed function-dependent semi-honest MPC protocol π between n parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$ using inputs $x_1, \ldots, x_n$ respectively. We first list some notation.

- Let $\mathsf{View}_{\mathcal{A}}^{k-1}(\{x_i\}_{i\in[n]})$ denote the view of a set of parties $\{\mathsf{P}_i\}_{i\in\mathcal{A}}$ after the first $(k-1)$ rounds of an execution of π.
- Let $\mathsf{View}_{\mathcal{A}}^{k}(\{x_i, \mathsf{state}_i\}_{i\in[n]}, \tau)$ denote the view of a set of parties $\{\mathsf{P}_i\}_{i\in\mathcal{A}}$ in the last round of an execution of π where τ is the transcript after $(k-1)$ rounds and for each party P_i, its input is x_i and its state is state_i.
- Also, let's assume that the size of any function is bounded by $p(\lambda)$ for some polynomial p.

Definition 14 (Delayed Function-Dependent and Reusable)**.** *Protocol π is said to be reusable if, for all $\mathcal{A} \subset [n]$ with $|\mathcal{A}| \leq (n-1)$, there exists a PPT simulator $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ such that for every PPT distinguisher $\mathcal{D}$, there exists a negligible function $\mu(\cdot)$ such that for all sufficiently large $\lambda \in \mathbb{N}$, the advantage of $\mathcal{D}$ is*

$$\mathsf{Adv}_{\mathcal{D}}^{\mathsf{Reus}} = \Big|\mathsf{Pr}[\mathsf{Expt}_{\mathcal{D}}^{\mathsf{Reus}}(1^\lambda, \{x_i\}_{i\in[n]}, \mathcal{A}, 0) = 1] - \mathsf{Pr}[\mathsf{Expt}_{\mathcal{D}}^{\mathsf{Reus}}(1^\lambda, \{x_i\}_{i\in[n]}, \mathcal{A}, 1) = 1]\Big| \leq \mu(\lambda),$$

where for each $b \in \{0,1\}$ and $\lambda \in \mathbb{N}$, the experiment $\mathsf{Expt}_{\mathcal{D}}^{\mathsf{Reus}}(1^\lambda, \{x_i\}_{i\in[n]}, \mathcal{A}, b)$ is defined below:

$\mathsf{Expt}_{\mathcal{D}}^{\mathsf{Reus}}(1^\lambda, \{x_i\}_{i\in[n]}, \mathcal{A}, b)$**:**

1. *If $b = 0$,* Challenger *runs the first $(k-1)$ rounds of π amongst n parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$ with inputs $x_1, \ldots, x_n$. Let state_i denote the state of party P_i after $(k-1)$ rounds.* Challenger *outputs $\mathsf{View}_{\mathcal{A}}^{k-1}(x_1, \ldots, x_n)$.*
2. *If $b = 1$,* Challenger *outputs $\mathsf{Sim}_1(p(\lambda), \{x_i\}_{i\in\mathcal{A}})$.*
3. Challenger *presents $\mathcal{D}$ with an oracle $\mathcal{O}_b$ that takes as input a function f and does the following:*
   (a) *If $b = 0$, execute the last round of protocol π for function f and output $\mathsf{View}_{\mathcal{A}}^{k}(\{x_i, \mathsf{state}_i\}_{i\in[n]}, \tau)$.*[5]
   (b) *If $b = 1$, output $(\mathsf{Sim}_2(f(x_1, \ldots, x_n), \{x_i\}_{i\in\mathcal{A}}))$.*

We show how to construct an unbounded-collusion secure FE combiner from such an MPC protocol. Formally, we show the following theorem:

Theorem 12. *Given any* delayed function-dependent *and* reusable *semi-honest MPC protocol π secure against all but one corruption, there exists an unbounded-collusion secure FE combiner.*

The two round semi-honest MPC protocol of Mukherjee and Wichs [MW16] based on the Learning With Errors (LWE) assumption is both *delayed function-dependent* and *reusable*. As a corollary of Theorem 12, we get:

Corollary 4. *Assuming LWE holds, there exists an unbounded-collusion secure FE combiner.*

The above corollary gives us a construction of an FE combiner from LWE that differs from the one in [AJS17].

[5]For simplicity, we assume that the function produces the same output for all parties and hence don't include the output along with the view.

7.1 Construction

We first list some notation for the construction.

Primitives and Notation Used:

- Let the n candidates be $\mathsf{FE}_1, \ldots, \mathsf{FE}_n$. Let the message space of the FE combiner be the set of all λ-bit strings.
- Let π denote any $k = \mathsf{poly}(\lambda)$ round semi-honest *delayed function-dependent* MPC protocol amongst $(n+1)$ parties where each party uses inputs of length λ. Let $p(\lambda)$ denote the sum of the length of the transcript of the protocol and the internal state of any party after the first $(k-1)$ rounds and let $q(\lambda)$ denote the length of the message sent by any party in the last round.
- In the above protocol π, let $\mathsf{Final.Round}_\pi(\cdot)$ denote the algorithm used by any party to compute its message in the last round and $\mathsf{Final.Output}_\pi(\cdot)$ denote the algorithm to compute the final output after the end of the k rounds. $\mathsf{Final.Round}_\pi$ takes as input the transcript up to $(k-1)$ rounds and the internal state of the party. Similarly, $\mathsf{Final.Output}_\pi$ takes as input the entire transcript of the protocol and the internal state of the party.
- Finally, let $\mathsf{Sym} = (\mathsf{Sym.Setup}, \mathsf{Sym.Enc}, \mathsf{Sym.Dec})$ denote any symmetric key encryption scheme with pseudorandom ciphertexts that encrypts messages of length $q(\lambda)$ using keys of length $p(\lambda)$ to generate ciphertexts of length $\ell(\lambda)$. Here, $p(\cdot), q(\cdot), \ell(\cdot)$ are all polynomials. We know that such a scheme can be built assuming one way functions e.g. using weak pseudorandom functions by defining $\mathsf{Sym.Enc}(\mathsf{K}, \mathsf{m}; \mathsf{r}) = (\mathsf{r}, \mathsf{PRF}_{\mathsf{K}}(\mathsf{r}) \oplus \mathsf{m})$ (see [Gol04] for more details).

Scheme:

We now describe the construction of our FE Combiner scheme FEComb.

$\mathsf{FEComb.Setup}(1^\lambda)$**:** The setup algorithm does the following:

1. For each $i \in [n]$, generate $\mathsf{MSK}_i \leftarrow \mathsf{FE}_i.\mathsf{Setup}(1^\lambda)$.
2. Output $\mathsf{MSK} = (\mathsf{MSK}_1, \ldots, \mathsf{MSK}_n)$.

$\mathsf{FEComb.Enc}(\mathsf{MSK}, x)$**:** Given input the master secret key MSK and a message x, the encryption algorithm does the following:

1. Choose λ-bit random strings $s_1, \ldots, s_{n-1}$ and set $s_n = (x \oplus s_1 \oplus \ldots \oplus s_{n-1})$.
2. Consider $(n+1)$ parties $\mathsf{P}_1, \ldots, \mathsf{P}_{n+1}$ where for all $i \in [n]$, P_i has input s_i and P_{n+1} has no input. Run the first $(k-1)$ rounds of protocol π amongst these parties. (Recall that this can be done independently of the function being computed)
3. After round $(k-1)$, let τ_x denote the transcript of the protocol and for each $i \in [n+1]$, let state_i denote the state of party P_i.
4. For each $i \in [n]$, compute $\mathsf{CT}_i = \mathsf{FE}_i.\mathsf{Enc}(\mathsf{MSK}_i, (\tau_x, \mathsf{state}_i, 0))$. Set $\mathsf{CT}_{n+1} = (\tau_x, \mathsf{state}_{n+1})$.
5. Output $\mathsf{CT} = (\mathsf{CT}_1, \ldots, \mathsf{CT}_{n+1})$.

$\mathsf{FEComb.Keygen}(\mathsf{MSK}, C)$**:** The setup algorithm does the following:

1. For each $i \in [n]$, choose a random string $\mathsf{Sym.CT}_i$ of length $\ell(\lambda)$ and generate $\mathsf{SK}_i \leftarrow \mathsf{FE}_i.\mathsf{KeyGen}(\mathsf{MSK}_i, C_i)$ where the circuit C_i is defined in Figure 4.

Input: $(\tau_x, \mathsf{state}_i, \beta)$
Hardwired: $\mathsf{Sym.CT}_i$

- If $\beta = 1$, set $k = (\tau_x, \mathsf{state}_i)$ and output $\mathsf{Sym.Dec}(k, \mathsf{Sym.CT}_i)$.
- Else, consider the party P_i that takes part in the MPC protocol π with n other parties to compute the circuit C. After $(k-1)$ rounds, let τ_x be the transcript of the protocol and state_i be P_i's internal state. Output $\mathsf{Final.Round}_\pi(\tau_x, \mathsf{state}_i)$.

Figure 4: Circuit C_i

2. Output $\mathsf{SK}_C = (\mathsf{SK}_1, \ldots, \mathsf{SK}_n)$.

$\underline{\mathsf{FEComb.Dec}(\mathsf{SK}_C, \mathsf{CT})}$**:** Given input a function secret key SK_C and a ciphertext CT, the decryption algorithm does the following:

1. Parse $\mathsf{SK}_C = (\mathsf{SK}_1, \ldots, \mathsf{SK}_n)$ and $\mathsf{CT} = (\mathsf{CT}_1, \ldots, \mathsf{CT}_n, \mathsf{CT}_{n+1})$.
2. For each $i \in [n]$, compute $\mathsf{out}_i = \mathsf{FE}_i.\mathsf{Dec}(\mathsf{SK}_i, \mathsf{CT}_i)$.
3. Parse $\mathsf{CT}_{n+1} = (\tau_x, \mathsf{state}_{n+1})$.
4. Now, consider τ_x to be the transcript after $(k-1)$ rounds of executing protocol π amongst n parties, $(\mathsf{out}_1, \ldots, \mathsf{out}_n)$ to be the messages of n parties in the last round and state_{n+1} to be the state of the other remaining party and compute its output.
5. That is, set $\mathsf{trans} = (\tau_x, \mathsf{out}_1, \ldots, \mathsf{out}_n)$ and output $\mathsf{Final.Output}_\pi(\mathsf{trans}, \mathsf{state}_{n+1})$.

Correctness: We observe that the correctness and polynomial slowdown properties follow by inspection.

7.2 Security Proof

In this section, we formally prove Theorem 11. The proof of Theorem 12 directly follows.

For the proof, let $(\mathsf{Sim}^1_\pi(\cdot), \mathsf{Sim}^2_\pi(\cdot))$ denote the simulator of the protocol π. Sim^1_π generates the messages of the first $(k-1)$ rounds, while Sim^2_π generates the last round message of the honest parties. Note that Sim^1_π only takes as input the adversary's input, while Sim^2_π also takes as input the output of the function.

To ease the exposition, let's consider the case where the adversary makes only one function key query. For an a priori bounded number of function key queries, we do the standard trick of repeating the scheme in parallel the bound number of times. We show that any PPT adversary $\mathcal{A}$ succeeds in the selective security game (Definition 15) with only negligible probability. We will show this via a sequence of hybrids $\mathsf{Hyb}_0, \ldots, \mathsf{Hyb}_4$. Let's say that candidate j is secure.

Hyb_0 **:** This hybrid corresponds to the real security game where the challenger picks a bit $b \in \{0, 1\}$ uniformly at random and computes $\mathsf{FEComb.Enc}(\mathsf{MSK}, x_b)$. Let's denote the challenge ciphertext by $\mathsf{CT}^* = (\mathsf{CT}^*_1, \ldots, \mathsf{CT}^*_{n+1})$.

Hyb_1 **:** In this hybrid, in the ciphertext generation step, the challenger picks $s_1, \ldots, s_{j-1}, s_{j+1}, \ldots, s_n$ uniformly at random and computes $s_j = (x \oplus s_1 \oplus \ldots \oplus s_{j-1} \oplus s_{j+1} \oplus \ldots \oplus s_n)$.

Hyb_2 **:** In this hybrid, the challenger first picks a key $\mathsf{Sym.SK} \leftarrow \mathsf{Sym.Setup}(1^\lambda)$. Then, for the function secret key query C made by the adversary, in order to generate the component SK_j, the challenger does the following:

- Recall the values $(\tau_x, \mathsf{state}_j)$ that were used to compute one of the components of the challenge ciphertext as follows: $\mathsf{CT}^*_j = \mathsf{FE}_j.\mathsf{Enc}(\mathsf{MSK}_j, (\tau_x, \mathsf{state}_j, 0)))$.
- Now, consider a party P_j that takes part in the MPC protocol π with n other parties and let τ_x be the transcript of the protocol, state_j be P_j's internal state after $(k-1)$ rounds.
- Compute the value $\mathsf{Sym.CT}_j \leftarrow \mathsf{Sym.Enc}(\mathsf{Sym.SK}, \mathsf{Final.Round}_\pi(\tau_x, \mathsf{state}_j))$.
- The rest of the procedure is exactly as in Hyb_0. That is, generate $\mathsf{SK}_j \leftarrow \mathsf{FE}_j.\mathsf{KeyGen}(\mathsf{MSK}_j, C_j)$ where the circuit C_j is defined in Figure 4 (with the hardwired value $\mathsf{Sym.CT}_j$ described above and replacing the index in the figure from i to j).

Hyb_3 **:** In this hybrid, the challenger generates the j^{th} component of the challenge ciphertext CT^*_j as follows: $\mathsf{CT}^*_j = \mathsf{FE}_j.\mathsf{Enc}(\mathsf{MSK}_j, (\mathsf{Sym.SK}, 1))$.

Hyb_4 **:** In this hybrid, the challenger computes the messages of party P_j (corresponding to the j^{th} candidate) in the challenge ciphertext and all the function secret keys using the simulator Sim_π of the MPC protocol π. More precisely, the challenger does the following:

- In the challenge ciphertext generation phase, s_j is picked randomly and no longer set to be $(x \oplus s_1 \oplus \ldots \oplus s_{j-1} \oplus s_{j+1} \oplus \ldots \oplus s_n)$.
- In the challenge ciphertext generation phase, to generate the transcript msg_x, compute the messages of party P_j by running the algorithm Sim^1_π. That is, treat the other n parties as corrupt parties with inputs $s_1, \ldots, s_n, \perp$ (except party P_j) and compute P_j's message as $\mathsf{Sim}^1_\pi(|C|, s_1, \ldots, s_{j-1}, s_{j+1}, \ldots, s_n, \perp)$.
- Then, for the function secret key query C, compute $\mathsf{Sym.CT}_j \leftarrow \mathsf{Sym.Enc}(\mathsf{Sym.SK}, \mathsf{Sim}^2_\pi(s_1, \ldots, s_{j-1}, s_{j+1}, \ldots, s_n, \perp, C(x))$ where $C(x)$ is the output of the protocol π executed by the $n+1$ parties.

We will now argue that every pair of consecutive hybrids are computationally indistinguishable and finally, show that any PPT adversary has negligible advantage in Hyb_4.

Lemma 16. Hyb_0 *is identical to* Hyb_1.

Proof. Since the only difference between the two hybrids is that in Hyb_0, s_n is computed as $(x \oplus s_1 \oplus \ldots \oplus s_{n-1})$ while in Hyb_1, s_j is computed as $(x \oplus s_1 \oplus \ldots \oplus s_{j-1} \oplus s_{j+1} \oplus \ldots \oplus s_n)$, it is easy to observe that they are identical. □

Lemma 17. *Assuming the pseudorandom ciphertexts property of* Sym, Hyb_1 *is computationally indistinguishable from* Hyb_2.

Proof. Suppose there exists an adversary $\mathcal{A}$ that can distinguish between the two hybrids with non-negligible probability. We will use $\mathcal{A}$ to construct an adversary $\mathcal{A}_{\mathsf{Sym}}$ that breaks the pseudorandom ciphertexts property of Sym which is a contradiction.

The adversary $\mathcal{A}_{\mathsf{Sym}}$ interacts with the adversary $\mathcal{A}$ as in Hyb_1. Then, on receiving the challenge message pair (x_0, x_1) and function secret key query C, to compute the function secret key, $\mathcal{A}_{\mathsf{Sym}}$ first sends the tuple $(\tau_x, \mathsf{state}_j)$ to the challenger $\mathcal{C}_{\mathsf{Sym}}$ of the scheme Sym. It receives a string which is either an encryption of this tuple using the scheme Sym or a uniformly random string. Then, $\mathcal{A}_{\mathsf{Sym}}$ sets this string to be the value CT^*_j in the challenge ciphertext and continues interacting with the adversary $\mathcal{A}$ exactly as in Hyb_1. Notice that when the challenger $\mathcal{C}_{\mathsf{Sym}}$ sends an actual ciphertext, the experiment between $\mathcal{A}_{\mathsf{Sym}}$ and $\mathcal{A}$ corresponds exactly to Hyb_1 and

when the challenger $\mathcal{C}_{\mathsf{Sym}}$ sends a uniformly random string, the experiment corresponds exactly to Hyb_2. Thus, if $\mathcal{A}$ can distinguish between the two hybrids with non-negligible probability, $\mathcal{A}_{\mathsf{Sym}}$ can use the same guess to break the pseudorandom ciphertexts property of the scheme Sym with non-negligible probability which is a contradiction. □

Lemma 18. *Assuming the security of the FE candidate* FE_j*,* Hyb_2 *is computationally indistinguishable from* Hyb_3*.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that can distinguish between the two hybrids with non-negligible probability. We will use $\mathcal{A}$ to construct an adversary $\mathcal{A}_{\mathsf{FE}}$ that breaks the security of the candidate FE_j which is a contradiction.

The adversary $\mathcal{A}_{\mathsf{FE}}$ interacts with the adversary $\mathcal{A}$ as in Hyb_2. Then, on receiving the challenge message pair (x_0, x_1) and function secret key query C, $\mathcal{A}_{\mathsf{FE}}$ first sends the pair of tuples $((\tau_x, \mathsf{state}_j, 0), (\mathsf{Sym.SK}, 1))$ as the challenge message pair along with function key query C_j (see Figure 4) to the challenger $\mathcal{C}_{\mathsf{FE}}$ of the FE candidate FE_i. Note that the function evaluates to the same value on both messages. From $\mathcal{C}_{\mathsf{FE}}$, $\mathcal{A}_{\mathsf{FE}}$ receives a ciphertext which is set as the value CT_j^* and a function secret key which is set as the value SK_j in its interaction with $\mathcal{A}$. Then, $\mathcal{A}_{\mathsf{FE}}$ continues interacting with the adversary $\mathcal{A}$ exactly as in Hyb_2.

Notice that when the challenger $\mathcal{C}_{\mathsf{FE}}$ encrypts the message $(\tau_x, \mathsf{state}_j, 0)$, the experiment between $\mathcal{A}_{\mathsf{FE}}$ and $\mathcal{A}$ corresponds exactly to Hyb_2 and when the challenger $\mathcal{C}_{\mathsf{FE}}$ encrypts the message $(\mathsf{Sym.SK}, 1)$, the experiment corresponds exactly to Hyb_3. Thus, if $\mathcal{A}$ can distinguish between the two hybrids with non-negligible probability, $\mathcal{A}_{\mathsf{FE}}$ can use the same guess to break the security of the candidate FE_j with non-negligible probability which is a contradiction. □

Lemma 19. *Assuming the security of the semi-honest* delayed function-dependent *MPC protocol* π*,* Hyb_3 *is computationally indistinguishable from* Hyb_4*.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that can distinguish between the two hybrids with non-negligible probability. We will use $\mathcal{A}$ to construct an adversary $\mathcal{A}_\pi$ that breaks the security of the MPC protocol π which is a contradiction.

The adversary $\mathcal{A}_\pi$ interacts with the adversary $\mathcal{A}$ as in Hyb_3 and receives the challenge message pair (x_0, x_1) and function secret key query C. Then, $\mathcal{A}_{\mathsf{FE}}$ interacts with the challenger in an MPC protocol π with $(n+1)$ parties where it corrupts parties $\mathsf{P}_1, \ldots, \mathsf{P}_{j-1}, \ldots, \mathsf{P}_{j+1}, \ldots, \mathsf{P}_{n+1}$. It receives a random input s_i for each party from $\mathcal{A}$ and interacts in the protocol receiving messages for party P_j from the challenger $\mathcal{C}_\pi$. $\mathcal{A}_\pi$ continues interacting with $\mathcal{A}$ exactly as in Hyb_3.

Notice that when the challenger $\mathcal{C}_\pi$ sends honestly generates messages for party P_j (real world), the experiment between $\mathcal{A}_\pi$ and $\mathcal{A}$ corresponds exactly to Hyb_3 and when the challenger $\mathcal{C}_\pi$ sends simulated messages for party P_j (ideal world), the experiment corresponds exactly to Hyb_4. Thus, if $\mathcal{A}$ can distinguish between the two hybrids with non-negligible probability, $\mathcal{A}_\pi$ can use the same guess to break the security of the MPC protocol π with non-negligible probability which is a contradiction. □

Lemma 20. *The adversary's advantage in* Hyb_4 *is negligible.*

Proof. Observe that the bit b is not used at all by the challenger in this hybrid and hence, any PPT adversary's advantage is negligible. □

Remark: Note that the proof of Theorem 12 directly follows. That is, to achieve unbounded collusion security, we will now rely on the reusability security property of the underlying MPC protocol.

8 From an FE Combiner to a Robust FE Combiner

The FE combiners constructed previously are not *robust*. By this, we mean that the constructions provide no guarantee of correctness or security if any of the underlying FE candidates do not satisfy correctness. However, determining the correctness of FE candidates may be difficult since a candidate FE may be correct with overwhelming probability on certain message, circuit pairs (m, C) and not others. With no worst-case guarantees, it can be challenging to reason about the correctness of an FE candidate especially if the function space $\mathcal{C}$ is say all poly-sized circuits, where sampling uniformly over the space is difficult.

We can mitigate this issue by making our FE combiners robust. A *robust FE combiner* is an FE combiner that satisfies correctness and security provided that at least one FE candidate, FE_i, satisfies both correctness and security. No restrictions are placed on the other FE candidates. In particular, they may satisfy neither correctness nor security. In this section, we show how to transform any FE combiner into a robust FE combiner. Formally, we show the following.

Theorem 13. *If there exists an FE combiner, then there exists a robust FE combiner.*

Combining Theorem 13 with Theorem 8, we obtain the following corollary.

Corollary 5. *Assuming the existence of a PRG in* NC^1*, there exists a robust FE combiner.*

This is done, at a high level, via the following steps.

1. Transform each FE candidate FE_i into a new FE candidate FE'_i such that
   (a) If FE_i is correct and secure, then FE'_i is also correct and secure.
   (b) If FE'_i is correct for any fixed message, circuit pair (m, C) with probability α, then it is at least α'-correct for all other message, circuit pairs (m', C') where $\alpha' = \alpha - \mathsf{negl}(\lambda)$.
2. Fix a message m and a circuit C and test each candidate repeatedly on (m, C) to determine if each candidate is α-correct for $\alpha \geq 1 - \frac{1}{\lambda}$. Discard those that are not.
3. Using standard techniques of BPP correctness amplification, transform the α-correct candidates into (almost) correct candidates.
4. Instantiate constructions of FE combiners from previous sections with these (almost) correct candidates.

8.1 FE Candidate Transformation

In this section, we describe the transformation described in step 1 in the construction overview in Section 8. The idea is to use a universal circuit U_m for a message m that takes as input the description of a circuit C and evaluates to $C(m)$. By encrypting a garbled circuit corresponding to U_m, we are able to transform correctness for a message, circuit pair (m, C) to correctness for a message, circuit pair where the message is a garbled circuit and the function is the evaluation circuit with the appropriate labels. We can then leverage the security of garbling scheme to argue that correctness of the new FE candidate cannot differ greatly between different message, circuit pairs. We now formally describe the transformation.

Let FE be an arbitrary FE candidate. Let $\mathsf{GC} = (\mathsf{Gen}, \mathsf{GrbC}, \mathsf{GrbI}, \mathsf{EvalGC})$ be a garbling scheme. Then, consider the following FE candidate FE'.

- $\mathsf{Setup}'(1^\lambda)$: Run $\mathsf{FE.Setup}(1^\lambda)$ to obtain MSK and $\mathsf{Gen}(1^\lambda)$ to obtain gcsk. Set $\mathsf{MSK}' = (\mathsf{MSK}, \mathsf{gcsk})$.

- $\mathsf{Enc}'(\mathsf{MSK}', m)$: Parse MSK' as $(\mathsf{MSK}, \mathsf{gcsk})$. Let U_m denote the universal circuit that takes as input a description $\langle C \rangle$ of a circuit $C \in \mathcal{C}$ and outputs $C(m)$. Let b be a uniformly random bit. Output
$$(\mathsf{FE.Enc}(\mathsf{MSK}, \mathsf{GrbC}(\mathsf{gcsk}, U_m(\cdot) \oplus b)), b)$$
as the ciphertext.

- $\mathsf{KeyGen}'(\mathsf{MSK}', C)$: Parse MSK' as $(\mathsf{MSK}, \mathsf{gcsk})$. Run $\mathsf{GrbI}(\mathsf{gcsk})$ to obtain $\vec{\mathbf{k}}$, the garbled circuit labels. Let $\vec{\mathbf{k}}(C)$ denote the labels corresponding to $\langle C \rangle$. Output
$$\mathsf{FE.KeyGen}(\mathsf{MSK}, \mathsf{EvalGC}(\cdot, \vec{\mathbf{k}}(C)))$$
as SK_C.

- $\mathsf{Dec}'(\mathsf{SK}_C, \mathsf{CT}')$: Parse CT' as (CT, b). Output
$$\mathsf{FE.Dec}(\mathsf{SK}_C, \mathsf{CT}) \oplus b.$$

Theorem 14. *If* FE *is a correct and secure FE candidate and* GC *is a garbling scheme, then* FE' *is also correct and secure.*

Proof. Correctness follows immediately from the correctness of FE and GC. In particular, fix any message m and circuit C. Let CT denote the encryption of m and SK_C denote the function key for C. Then,
$$\begin{aligned}\mathsf{FE}'.\mathsf{Dec}(\mathsf{SK}_C, \mathsf{CT}) &= \mathsf{EvalGC}(\mathsf{GrbC}(\mathsf{gcsk}, U_m(\cdot) \oplus b), \vec{\mathbf{k}}(C)) \oplus b \\ &= U_m(\langle C \rangle) \oplus b \oplus b = C(m).\end{aligned}$$

For security, suppose there exists an adversary $\mathcal{A}$ that can win at the indistinguishability security game for FE'. Then, consider the adversary $\mathcal{A}'$ that breaks the security of FE. $\mathcal{A}'$ runs $\mathcal{A}$ and plays the role of the challenger. When $\mathcal{A}$ submits a message pair (m_0, m_1), $\mathcal{A}'$ runs $\mathsf{Gen}(1^\lambda)$ to obtain gcsk. Then, it obtains $m_0' = \mathsf{GrbC}(\mathsf{gcsk}, U_{m_0} \oplus b)$ and similarly for m_1' for a uniformly sampled bit b. It then submits (m_0', m_1') to its challenger to receive CT and gives (CT, b) to $\mathcal{A}$. Whenever, $\mathcal{A}$ requests a function key for a circuit C, $\mathcal{A}'$ runs $\mathsf{GrbI}(\mathsf{gcsk})$ to obtain $\vec{\mathbf{k}}$ and then requests a function key to its challenger for the circuit $\mathsf{EvalGC}(\cdot, \vec{\mathbf{k}}(C))$. It gives the resulting function key to $\mathcal{A}$. When $\mathcal{A}$ outputs a response, $\mathcal{A}'$ outputs the same response. Note that $\mathcal{A}'$ simulates the security game for $\mathcal{A}$ perfectly and wins whenever $\mathcal{A}$ wins. Therefore, $\mathcal{A}'$ is a distinguisher for FE, contradicting the security of FE. □

Theorem 15. *Let* FE' *be the result of applying the above transformation to an FE candidate* FE *and let* GC *be a garbling scheme. For a message, circuit pair* (m, C)*, let*
$$\Pr[\mathsf{Corr}(m, C)] = \Pr\left[\begin{array}{c} \mathsf{MSK} \leftarrow \mathsf{FE}'.\mathsf{Setup}(1^\lambda) \\ \mathsf{CT} \leftarrow \mathsf{FE}'.\mathsf{Enc}(\mathsf{MSK}, m) \\ \mathsf{SK}_C \leftarrow \mathsf{FE}'.\mathsf{KeyGen}(\mathsf{MSK}, C) \\ C(m) \leftarrow \mathsf{FE}'.\mathsf{Dec}(\mathsf{SK}_C, \mathsf{CT}) \end{array}\right]$$
denote the probability of correctness when FE' *is run on* (m, C)*. Then, for any two message, circuit pairs* (m_0, C_0) *and* (m_1, C_1)*,*
$$|\Pr[\mathsf{Corr}(m_0, C_0)] - \Pr[\mathsf{Corr}(m_1, C_1)]| \leq \mathsf{negl}(\lambda).$$

Proof. We will show this via a series of experiments. Fix any message, circuit pair (m, C).

$\mathsf{Expt}_0(1^\lambda, m, C)$:

1. Run $\mathsf{MSK}' \leftarrow \mathsf{FE}'.\mathsf{Setup}(1^\lambda)$.
2. Run $\mathsf{CT}' \leftarrow \mathsf{FE}'.\mathsf{Enc}(\mathsf{MSK}', m)$.
3. Run $\mathsf{SK}_C \leftarrow \mathsf{FE}'.\mathsf{KeyGen}(\mathsf{MSK}', C)$.
4. If $\mathsf{FE}'.\mathsf{Dec}(\mathsf{SK}_C, \mathsf{CT}') = C(m)$, output 1. Else, output 0.

$\mathsf{Expt}_1(1^\lambda, m, C)$:

1. Run $(\mathsf{MSK}, \mathsf{gcsk}) \leftarrow \mathsf{FE}'.\mathsf{Setup}(1^\lambda)$.
2. Sample a uniformly random bit b. Let C' denote the circuit $U_m(\cdot) \oplus b$.
3. Run $\mathsf{SimGC}(1^\lambda, \phi(C'), C(m) \oplus b)$ to obtain $(\widehat{C'}, k_1, \ldots, k_\ell)$.
4. Set $\mathsf{CT} = \mathsf{FE.Enc}(\mathsf{MSK}, \widehat{C'})$.
5. Set $\mathsf{SK}_C = \mathsf{FE.KeyGen}(\mathsf{MSK}, \mathsf{EvalGC}(\cdot, (k_1, \ldots, k_\ell)))$.
6. If $\mathsf{FE.Dec}(\mathsf{SK}_C, \mathsf{CT}) \oplus b = C(m)$, output 1. Else, output 0.

$\mathsf{Expt}_2(1^\lambda, m, C)$:

1. Run $(\mathsf{MSK}, \mathsf{gcsk}) \leftarrow \mathsf{FE}'.\mathsf{Setup}(1^\lambda)$.
2. Sample a uniformly random bit b'. Let $\phi(C')$ denote the topology of the circuit $U_m(\cdot) \oplus b$. This is independent of m and b.
3. Run $\mathsf{SimGC}(1^\lambda, \phi(C'), b')$ to obtain $(\widehat{C'}, k_1, \ldots, k_\ell)$.
4. Set $\mathsf{CT} = \mathsf{FE.Enc}(\mathsf{MSK}, \widehat{C'})$.
5. Set $\mathsf{SK}_C = \mathsf{FE.KeyGen}(\mathsf{MSK}, \mathsf{EvalGC}(\cdot, (k_1, \ldots, k_\ell)))$.
6. If $\mathsf{FE.Dec}(\mathsf{SK}_C, \mathsf{CT}) = b'$, output 1. Else, output 0.

Lemma 21.

$$\left|\Pr\left[\mathsf{Expt}_0(1^\lambda, m, C) = 1\right] - \Pr\left[\mathsf{Expt}_1(1^\lambda, m, C) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

Proof. Suppose the above did not hold. Then, consider the following adversary $\mathcal{A}$ that breaks the security of GC. $\mathcal{A}$ samples a random bit b and submits its challenge circuit to be $U_m(\cdot) \oplus b$ and its challenge input to be $\langle C \rangle$. It receives a garbled circuit $\widehat{C'}$ and wire keys $k_1, \ldots, k_\ell$. $\mathcal{A}$ runs $\mathsf{FE.Setup}(1^\lambda)$ to obtain MSK. It then sets $\mathsf{CT} = \mathsf{FE.Enc}(\mathsf{MSK}, \widehat{C'})$. It additionally sets $\mathsf{SK}_C = \mathsf{FE.KeyGen}(\mathsf{MSK}, \mathsf{EvalGC}(\cdot, (k_1, \ldots, k_\ell)))$. It then checks if $\mathsf{FE.Dec}(\mathsf{SK}_C, \mathsf{CT}) \oplus b = C(m)$. If it does, it outputs 1. Otherwise, it outputs 0. Note that if $\mathcal{A}$'s challenger is using the garbling scheme, then $\mathcal{A}$ runs $\mathsf{Expt}_0(1^\lambda, m, C)$. On the other hand, if $\mathcal{A}$'s challenger is using the garbled circuit simulator, then $\mathcal{A}$ runs $\mathsf{Expt}_1(1^\lambda, m, C)$. So, if the lemma did not hold, $\mathcal{A}$ would break the security of GC. □

Lemma 22.

$$\left|\Pr\left[\mathsf{Expt}_1(1^\lambda, m, C) = 1\right] - \Pr\left[\mathsf{Expt}_2(1^\lambda, m, C) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

Proof. This follows immediately from the definitions of the experiments. Note that these two experiments are functionally equivalent, since the distributions of a random bit b' and $C(m) \oplus b$ for a random bit b are identical. Additionally, $\mathsf{FE.Dec}(\mathsf{SK}_C, \mathsf{CT}) \oplus b = C(m)$ if and only if $\mathsf{FE.Dec}(\mathsf{SK}_C, \mathsf{CT}) = C(m) \oplus b$, which is distributed the same as a random bit b'. Therefore, the lemma holds.

Now, note that $\mathsf{Expt}_2(1^\lambda, m, C)$ is independent of the message, circuit pair (m, C) since it depends only on the topology of the circuit $U_m(\cdot) \oplus b$, which is the same regardless of m. Since $\mathsf{Expt}_0(1^\lambda, m, C)$ is 1 with the same probability as the probability statement in the theorem, it follows from the above lemmas that for any message, circuit pairs (m_0, C_0) and (m_1, C_1), that

$$\left|\Pr\left[\mathsf{Expt}_0(1^\lambda, m_0, C_0) = 1\right] - \Pr\left[\mathsf{Expt}_2(1^\lambda, m_0, C_0) = 1\right]\right| \leq \mathsf{negl}(\lambda)$$

and

$$\left|\Pr\left[\mathsf{Expt}_0(1^\lambda, m_1, C_1) = 1\right] - \Pr\left[\mathsf{Expt}_2(1^\lambda, m_1, C_1) = 1\right]\right| \leq \mathsf{negl}(\lambda)$$

and

$$\left|\Pr\left[\mathsf{Expt}_2(1^\lambda, m_0, C_0) = 1\right] - \Pr\left[\mathsf{Expt}_2(1^\lambda, m_1, C_1) = 1\right]\right| \leq \mathsf{negl}(\lambda)$$

and so the theorem holds. □

8.2 Proof of Theorem 13

Using the FE candidate transformation described in Section 8.1, we can now show Theorem 13. Given FE candidates $\{\mathsf{FE}_i\}_i$ with the guarantee that at least one is correct and secure, run the FE transformation from Section 8.1 to obtain new FE candidates $\{\mathsf{FE}'_i\}_i$. Next, fix an arbitrary message m and circuit C and test for correctness on (m, C) a total of λ^2 times. If the FE candidate is not correct on any of the tests, discard it. By a Chernoff bound, all remaining candidates will be α-correct for $\alpha \geq 1 - 1/\lambda$ with overwhelming probability. Note that since the secure FE candidate must be (almost) correct, it will not be discarded with overwhelming probability. Next, use standard techniques of BPP amplification (repeat in parallel a $\mathsf{poly}(\lambda)$ number of times and take a majority vote) to transform all the remaining FE candidates into (almost) correct candidates. This follows from [AJN+16]. Finally, instantiate the FE combiner construction from Section 6 to obtain a robust FE combiner.

Universal Functional Encryption: Robust FE combiners are closely related to the notion of universal functional encryption. Universal functional encryption is a construction of functional encryption satisfying the following simple guarantee. If there exists a Turing Machine with running time bounded by some $T(n) = \mathsf{poly}(n)$ that implements a correct and secure FE scheme, then the universal functional encryption construction is itself a correct and secure FE scheme. Using the existence of a robust FE combiner (Theorem 13) and the results of [AJN+16], we observe the following.

Theorem 16. *Assuming the existence of a robust FE combiner, there exists a universal functional encryption scheme.*

Using the above theorem and Corollary 5, we arrive at the following corollary.

Corollary 6. *Assuming the existence of a PRG in* NC^1*, there exists a universal functional encryption scheme.*

References

[AB81] C.A. Asmuth and G.R. Blakley. An efficient algorithm for constructing a cryptosystem which is harder to break than two other cryptosystems. *Computers and Mathematics with Applications*, 1981.

[ABSV15] Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. From selective to adaptive security in functional encryption. In *CRYPTO*, 2015.

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

[AIK05] B. Applebaum, Y. Ishai, and E. Kushilevitz. Computationally private randomizing polynomials and their applications (extended abstract). In *CCC*, June 2005.

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In *CRYPTO*, 2015.

[AJLA+12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold fhe. In *EUROCRYPT*, 2012.

[AJN+16] Prabhanjan Ananth, Aayush Jain, Moni Naor, Amit Sahai, and Eylon Yogev. Universal constructions and robust combiners for indistinguishability obfuscation and witness encryption. In *CRYPTO*, 2016.

[AJS15] Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Indistinguishability obfuscation from functional encryption for simple functions. Cryptology ePrint Archive, Report 2015/730, 2015.

[AJS17] Prabhanjan Ananth, Aayush Jain, and Amit Sahai. Robust transforming combiners from indistinguishability obfuscation to functional encryption. In *EUROCRYPT*, 2017.

[AS17] Prabhanjan Ananth and Amit Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In *EUROCRYPT*, 2017.

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In *EUROCRYPT*, 2014.

[BGG+17] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M.R. Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2017, 2017.

[BGI+18] Elette Boyle, Niv Gilboa, Yuval Ishai, Huijia Lin, and Stefano Tessaro. Foundations of homomorphic secret sharing. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 94. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[BGJS17] Saikrishna Badrinarayanan, Vipul Goyal, Aayush Jain, and Amit Sahai. A note on vrfs from verifiable functional encryption. *IACR Cryptology ePrint Archive*, 2017:51, 2017.

[BGV14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Trans. Comput. Theory*, 6(3):13:1–13:36, July 2014.

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *CCS*, 2012.

[Bit17] Nir Bitansky. Verifiable random functions from non-interactive witness-indistinguishable proofs. In *TCC*, 2017.

[BL18] Fabrice Benhamouda and Huijia Lin. k-round mpc from k-round ot via garbled interactive circuits. *EUROCRYPT*, 2018.

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols. In *STOC*, 1990.

[BOGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *STOC*, 1988.

[BP16] Zvika Brakerski and Renen Perlman. Lattice-based fully dynamic multi-key fhe with short ciphertexts. In *CRYPTO*, pages 190–213. Springer, 2016.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In *EUROCRYPT*, 2012.

[BS03] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324(1):71–90, 2003.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography*, pages 253–273. Springer, 2011.

[BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In *FOCS*, 2015.

[BV16] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation: from approximate to exact. In *Theory of Cryptography Conference*, pages 67–95. Springer, 2016.

[BV17] Nir Bitansky and Vinod Vaikuntanathan. A note on perfect correctness by derandomization. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 592–606. Springer, 2017.

[CGH+15] Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrede Lepoint, Hemanta K Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. Zeroizing without low-level zeroes: New mmap attacks and their limitations. In *CRYPTO*, 2015.

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

[CJL16] Jung Hee Cheon, Jinhyuck Jeong, and Changmin Lee. An algorithm for cspr problems and cryptanalysis of the ggh multilinear map without an encoding of zero. Technical report, Cryptology ePrint Archive, Report 2016/139, 2016.

[CLLT15] Jean-Sebastien Coron, Moon Sung Lee, Tancrede Lepoint, and Mehdi Tibouchi. Cryptanalysis of ggh15 multilinear maps. Cryptology ePrint Archive, Report 2015/1037, 2015.

[CM15] Michael Clear and Ciaran McGoldrick. Multi-identity and multi-key leveled fhe from learning with errors. In *CRYPTO*, 2015.

[FHNS16] Marc Fischlin, Amir Herzberg, Hod Bin Noon, and Haya Shulman. Obfuscation combiners. In *CRYPTO*, 2016.

[GGH13] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, pages 1–17, 2013.

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

[GHKW17] Rishab Goyal, Susan Hohenberger, Venkata Koppula, and Brent Waters. A generic approach to constructing and proving verifiable random functions. In *TCC*, pages 537–566. Springer, 2017.

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

[GKW17] Shafi Goldwasser, Saleet Klein, and Daniel Wichs. The edited truth. In *TCC*, pages 305–340. Springer, 2017.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game. In *STOC*, pages 218–229. ACM, 1987.

[Gol04] Oded Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.

[GPS16] Sanjam Garg, Omkant Pandey, and Akshayaram Srinivasan. Revisiting the cryptographic hardness of finding a nash equilibrium. In *CRYPTO*, 2016.

[GPSZ17] Sanjam Garg, Omkant Pandey, Akshayaram Srinivasan, and Mark Zhandry. Breaking the sub-exponential barrier in obfustopia. In *EUROCRYPT*, 2017.

[GS17] Sanjam Garg and Akshayaram Srinivasan. Garbled protocols and two-round mpc from bilinear maps. *FOCS*, 2017.

[GS18] Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. *EUROCRYPT*, 2018.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, 2013.

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

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In *CRYPTO*, 2015.

[Her05] Amir Herzberg. On tolerant cryptographic constructions. In *CT-RSA*, 2005.

[Her09] Amir Herzberg. Folklore, practice and theory of robust combiners. *Journal of Computer Security*, 17(2):159–189, 2009.

[HJ15] Yupu Hu and Huiwen Jia. Cryptanalysis of ggh map. *IACR Cryptology ePrint Archive*, 2015:301, 2015.

[HJO+16] Brett Hemenway, Zahra Jafargholi, Rafail Ostrovsky, Alessandra Scafuro, and Daniel Wichs. Adaptively secure garbled circuits from one-way functions. In *CRYPTO*, 2016.

[HKN+05] Danny Harnik, Joe Kilian, Moni Naor, Omer Reingold, and Alon Rosen. On robust combiners for oblivious transfer and other primitives. In *EUROCRYPT*, 2005.

[IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 21–30. ACM, 2007.

[Lev87] Leonid A. Levin. One-way functions and pseudorandom generators. *Combinatorica*, 1987.

[Lin16] Huijia Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In *EUROCRYPT*, 2016.

[Lin17] Huijia Lin. Indistinguishability obfuscation from sxdh on 5-linear maps and locality-5 prgs. In *CRYPTO*, 2017.

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

[LPST16a] Huijia Lin, Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation with non-trivial efficiency. In *PKC*, 2016.

[LPST16b] Huijia Lin, Rafael Pass, Karn Seth, and Sidharth Telang. Output-compressing randomized encodings and applications. In *TCC*, 2016.

[LT17] Huijia Lin and Stefano Tessaro. Indistinguishability obfuscation from trilinear maps and block-wise local prgs. In *CRYPTO*, 2017.

[LV16] Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from ddh-like assumptions on constant-degree graded encodings. In *FOCS*, 2016.

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key fhe. In *EUROCRYPT*, 2016.

[NR04] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *Journal of the ACM (JACM)*, 51(2):231–262, 2004.

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

[PS16] Chris Peikert and Sina Shiehian. Multi-key fhe from lwe, revisited. In *TCC*, 2016.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO*, 2008.

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

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

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

A Additional Preliminaries

A.1 Functional Encryption: Security Definitions

IND-Security. We recall indistinguishability-based selective security for FE. This security notion is modeled as a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ where the adversary can request functional keys and ciphertexts from $\mathcal{C}$. Specifically, $\mathcal{A}$ can submit function queries C and $\mathcal{C}$ responds with the corresponding functional keys. $\mathcal{A}$ can also submit message queries of the form (x_0, x_1) and receives an encryption of messages x_b for some bit $b \in \{0,1\}$. The adversary $\mathcal{A}$ wins the game if she can guess b with probability significantly more than $1/2$ and if for all function queries C and message queries (x_0, x_1), $C(x_0) = C(x_1)$. That is to say, any function evaluation that is computable by $\mathcal{A}$ gives the same value regardless of b. It is required that the adversary must declare the challenge messages at the beginning of the game.

Definition 15 (IND-secure FE)**.** *A secret-key FE scheme* FE *for a class of circuits* $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in[\mathbb{N}]}$ *and message space* $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in[\mathbb{N}]}$ *is selectively secure if for any PPT adversary* $\mathcal{A}$*, there exists a negligible function* $\mu(\cdot)$ *such that for all sufficiently large* $\lambda \in \mathbb{N}$*, the advantage of* $\mathcal{A}$ *is*

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

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

1. ***Challenge message queries****:* $\mathcal{A}$ *submits message queries,*

$$\left\{(x_0^i, x_1^i)\right\}$$

with $x_0^i, x_1^i \in \mathcal{X}_\lambda$ *to the challenger* $\mathcal{C}$*.*

2. $\mathcal{C}$ *computes* $\mathsf{MSK} \leftarrow \mathsf{FE.Setup}(1^\lambda)$ *and then computes* $\mathsf{CT}_i \leftarrow \mathsf{FE.Enc}(\mathsf{MSK}, x_b^i)$ *for all* i*. The challenger* $\mathcal{C}$ *then sends* $\{\mathsf{CT}_i\}$ *to the adversary* $\mathcal{A}$*.*

3. ***Function queries****: The following is repeated an at most polynomial number of times:* $\mathcal{A}$ *submits a function query* $C \in \mathcal{C}_\lambda$ *to* $\mathcal{C}$*. The challenger* $\mathcal{C}$ *computes* $\mathsf{SK}_C \leftarrow \mathsf{FE.KeyGen}(\mathsf{MSK}, C)$ *and sends it to* $\mathcal{A}$*.*

4. *If there exists a function query* C *and challenge message queries* (x_0^i, x_1^i) *such that* $C(x_0^i) \neq C(x_1^i)$*, then the output of the experiment is set to* $\perp$*. Otherwise, the output of the experiment is set to* b'*, where* b' *is the output of* $\mathcal{A}$*.*

Adaptive Security. The above security notion is referred to as selective security in the literature. One can consider a stronger notion of security, called *adaptive security*, where the adversary can interleave the challenge messages and the function queries in any arbitrary order. Analogous to Definition 15, we can define an adaptively secure FE scheme. In this paper, we only deal with selectively secure FE schemes. However, the security of these schemes can be upgraded to adaptive with no additional cost [ABSV15].

Simulation Security. We can also consider a different notion of security, called (single-key) simulation security.

Definition 16. *(SIM-Security) Let* FE *denote a functional encryption scheme for a circuit class* $\mathcal{C}$*. For every PPT adversary* $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ *and a PPT simulator* Sim*, consider the following two experiments:*

$\mathsf{Exp}^{\mathsf{real}}_{\mathsf{FE},\mathsf{A}}(1^\lambda)$	$\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{FE},\mathsf{A},\mathsf{Sim}}(1^\lambda)$
$\{\mathsf{FE.Setup}(1^\lambda) \to \mathsf{MSK}\}$	$\{\mathsf{FE.Setup}(1^\lambda) \to \mathsf{MSK}\}$
$\mathsf{A}_1 \to (C, \mathsf{state}_{\mathsf{A}_1})$	$\mathsf{A}_1 \to (C, \mathsf{state}_{\mathsf{A}_1})$
$\{\mathsf{SK}_C \leftarrow \mathsf{FE.KeyGen}(\mathsf{MSK}, C)\}$	$\{\mathsf{SK}_C \leftarrow \mathsf{FE.KeyGen}(\mathsf{MSK}, C)\}$
$\mathsf{A}_2(\mathsf{state}_{\mathsf{A}_1}, \mathsf{SK}_C) \to (m, \mathsf{state}_{\mathsf{A}_2})$	$\mathsf{A}_2(\mathsf{state}_{\mathsf{A}_1}, \mathsf{SK}_C) \to (m, \mathsf{state}_{\mathsf{A}_2})$
$\mathsf{FE.Enc}(\mathsf{MSK}, m) \to \mathsf{CT}$	$\mathsf{Sim}(\mathsf{MSK}, C, \mathsf{SK}_C, C(m)) \to \mathsf{CT}$
Output $(\mathsf{CT}, \mathsf{state}_{\mathsf{A}_2})$	*Output* $(\mathsf{CT}, \mathsf{state}_{\mathsf{A}_2})$

The scheme is said to be (single-key) SIM-secure if there exists a PPT simulator Sim *such that for all PPT adversaries* $(\mathsf{A}_1, \mathsf{A}_2)$*, the outcomes of the two experiments are computationally indistinguishable:*

$$\{\mathsf{Exp}^{\mathsf{real}}_{\mathsf{FE},\mathsf{A}}(1^\lambda)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{FE},\mathsf{A},\mathsf{Sim}}(1^\lambda)\}_{\lambda\in\mathbb{N}}$$

A.2 Secure Multi-Party Computation

We now provide the necessary background on secure multi-party computation. We first present the syntax and then the security definition.

Syntax. We define a secure multi-party computation protocol Π for n parties $P_1, \ldots, P_n$ associated with an n-party functionality $f : \{0,1\}^{\ell_1} \times \cdots \times \{0,1\}^{\ell_n} \to \{0,1\}^{\ell_{y_1}} \times \cdots \times \{0,1\}^{\ell_{y_n}}$. We let ℓ_i denote the length of the i^{th} party's input and ℓ_{y_i} denote the length of the i^{th} party's output. In any given execution of the protocol, the i^{th} party receives as input $x_i \in \{0,1\}^{\ell_i}$ and maintains a state state_i. All the parties jointly compute the functionality $f(x_1, \ldots, x_n)$ by following the protocol. The protocol proceeds in rounds and in each round, every party broadcasts a message to the other parties. In the end, party P_i outputs out_i. The protocol is said to be correct if the joint distribution $(\mathsf{out}_1, \ldots, \mathsf{out}_n)$ is statistically close to $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$.

Semi-Honest Adversaries. We consider weaker adversarial models where the adversaries follow the instructions of the protocol. That is, they receive their inputs from the environment, behave as prescribed by the protocol, and, finally, output their view of the protocol. Such types of adversaries are referred to as semi-honest adversaries.

We define semi-honest security below. Denote $\mathsf{Real}^{\Pi}_{f,S}(x_1, \ldots, x_n)$ to be the joint distribution over the outputs of all the parties along with the views of the parties indexed by the set S.

Definition 17 (Semi-Honest Security)**.** *Consider an n-party functionality f as defined above. Fix a set of inputs $(x_1, \ldots, x_n)$, where $x_i \in \{0,1\}^{\ell_i}$ and let r_i be the randomness of the i^{th} party. Let Π be a n-party protocol implementing f. We say that Π satisfies* **security against semi-honest adversaries** *if for every subset of parties S, there exists a PPT simulator Sim such that:*

$$\{\ (\{y_i\}_{i\notin S}, \mathsf{Sim}\left(\{y_i\}_{i\in S}, \{x_i\}_{i\in S}\right))\ \} \cong_c \left\{\mathsf{Real}^{\Pi}_{f,S}(x_1, \ldots, x_n)\right\},$$

where y_i is the i^{th} output of $f(x_1, \ldots, x_n)$. In particular, we say that Π is semi-honest secure.

A.3 Threshold Leveled Fully Homomorphic Encryption

The following definition of threshold homomorphic encryption is adapted from [AJLA+12, MW16, BGG+17]. A threshold homomorphic encryption scheme is a tuple of PPT algorithms $\mathsf{TFHE} = (\mathsf{TFHE.Setup}, \mathsf{TFHE.Enc}, \mathsf{TFHE.Eval}, \mathsf{TFHE.PartDec}, \mathsf{TFHE.FinDec})$ satisfying the following specifications:

- **Setup,** $\mathsf{Setup}(1^\lambda, 1^d, 1^n)$: It takes as input the security parameter λ, a circuit depth d, and the number of parties n. It outputs a public key fpk and secret key shares $\mathsf{fsk}_1, \ldots, \mathsf{fsk}_n$.
- **Encryption,** $\mathsf{Enc}(\mathsf{fpk}, \mu)$: It takes as input a public key fpk and a single bit plaintext $\mu \in \{0,1\}$ and outputs a ciphertext CT.
- **Evaluation,** $\mathsf{Eval}(C, \mathsf{CT}_1, \ldots, \mathsf{CT}_k)$: It takes as input a boolean circuit $C\colon \{0,1\}^k \to \{0,1\} \in \mathcal{C}_\lambda$ of depth $\leq d$ and ciphertexts $\mathsf{CT}_1, \ldots, \mathsf{CT}_k$ encrypted under the same public key. It outputs an evaluation ciphertext CT. We shall assume that the ciphertext also contains fpk.
- **Partial Decryption,** $p_i \leftarrow \mathsf{PartDec}(\mathsf{fsk}_i, \mathsf{CT})$: It takes as input a secret key share fsk_i and a ciphertext CT. It outputs a partial decryption p_i related to the party i.
- **Final Decyrption,** $\mathsf{FinDec}(B)$: It is a deterministic algorithm that takes as input a set $B = \{p_i\}_{i\in[n]}$. It outputs a plaintext $\widehat{\mu} \in \{0, 1, \perp\}$.

Definition 18 (TFHE)**.** *A TFHE scheme is required to satisfy the following properties for all parameters $(\mathsf{fpk}, \mathsf{fsk}_1, \ldots, \mathsf{fsk}_N) \leftarrow \mathsf{Setup}(1^\lambda, 1^d, 1^n)$, any plaintexts $\mu_1, \ldots, \mu_k \in \{0,1\}$, and any boolean circuit $C\colon \{0,1\}^k \to \{0,1\} \in \mathcal{C}_\lambda$ of depth $\leq d$.*

Correctness of Encryption. *Let $\mathsf{CT} = \mathsf{Enc}(\mathsf{fpk}, \mu_1)$ and $B = \{\mathsf{PartDec}(\mathsf{fsk}_i, \mathsf{CT})\}_{i\in[n]}$. With all but negligible probability in λ over the coins of Setup, Enc, and $\mathsf{PartDec}$, $\mathsf{FinDec}(B) = \mu_1$.*

Correctness of Evaluation. *Let $\mathsf{CT}_i = \mathsf{Enc}(\mathsf{fpk}, \mu_i)$ for $1 \leq i \leq k$, $\widehat{\mathsf{CT}} = \mathsf{Eval}(C, \mathsf{CT}_1, \ldots, \mathsf{CT}_k)$, and $B = \{\mathsf{PartDec}(\mathsf{fsk}_i, \widehat{\mathsf{CT}},)\}_{i\in[n]}$. With all but negligible probability in λ over the coins of Setup, Enc, and $\mathsf{PartDec}$, $\mathsf{FinDec}(B) = C(\mu_1, \ldots, \mu_k)$.*

Compactness of Ciphertexts. *There exists a polynomial, poly, such that $|\mathsf{CT}| \leq \mathsf{poly}(\lambda, d)$ for any ciphertext CT generated from the algorithms of TFHE.*

Semantic Security of Encryption. *Any PPT adversary $\mathcal{A}$ has only negligible advantage as a function of λ over the coins of all the algorithms in the following game:*

1. *Run $\mathsf{Setup}(1^\lambda, 1^d, 1^n) \to (\mathsf{fpk}, \mathsf{fsk}_1, .., \mathsf{fsk}_n)$. The adversary is given fpk.*
2. *The adversary outputs a set $S \subset [n]$ of size $n - 1$.*
3. *The adversary receives $\{\mathsf{fsk}_i\}_{i\in S}$ along with $\mathsf{Enc}(\mathsf{fpk}, b) \to \mathsf{CT}$ for a random $b \in \{0,1\}$.*

4. *The adversary outputs b' and wins if $b = b'$.*

Simulation Security. *Let $\mathsf{CT}_i = \mathsf{Enc}(\mathsf{fpk}, \mu_i)$ for $1 \le i \le k$, $\hat{\mathsf{CT}} = \mathsf{Eval}(C, \mathsf{CT}_1, \ldots, \mathsf{CT}_k)$, and $p_i = \mathsf{PartDec}(\mathsf{fsk}_i, \hat{\mathsf{CT}},)$ for all $i \in [n]$. There exists a PPT algorithm Sim such that for any subset S of the form $[n] \setminus i^*$, $\mathsf{Sim}(\hat{\mathsf{CT}}, \{\mathsf{fsk}\}_S, C(\mu_1, .., \mu_k)) \to p'_{i^*}$ the following distributions are statistically close (in the security parameter):*

$$(p_i, \mathsf{fpk}, \mathsf{CT}_1, .., \mathsf{CT}_k, \{\mathsf{fsk}_i\}_{i\in[n]}) \approx (p'_{i^*}, \mathsf{fpk}, \mathsf{CT}_1, .., \mathsf{CT}_k, \{\mathsf{fsk}_i\}_{i\in[n]}).$$

A.4 Garbling Schemes

We recall the definition of garbling schemes [Yao86, BHR12].

Definition 19 (Garbling Schemes [Yao86, BHR12])**.** *A garbling scheme $\mathsf{GC} = (\mathsf{Gen}, \mathsf{Grbl}, \mathsf{GrbC}, \mathsf{EvalGC})$ defined for a class of circuits $\mathcal{C}$ consists of the following polynomial time algorithms:*

- **Setup,** $\mathsf{Gen}(1^\lambda)$*: On input security parameter λ, it generates the secret parameters* gcsk*.*
- **Generation of Garbling Keys,** $\mathsf{Grbl}(\mathsf{gcsk})$*: On input secret parameters* gcsk*, it generates the wire keys $\vec{\mathbf{k}} = (\mathbf{k}_1, \ldots, \mathbf{k}_\ell)$, where $\mathbf{k}_i = (k_i^0, k_i^1)$.*
- **Garbled Circuit Generation,** $\mathsf{GrbC}(\mathsf{gcsk}, C)$*: On input secret parameters* gcsk *and circuit $C \in \mathcal{C}$, it generates the garbled circuit $\widehat{C}$.*
- **Evaluation,** $\mathsf{EvalGC}(\widehat{C}, (k_1^{x_1}, \ldots, k_\ell^{x_\ell}))$*: On input garbled circuit $\widehat{C}$, wire keys $(k_1^{x_1}, \ldots, k_\ell^{x_\ell})$, it generates the output* out*.*

It satisfies the following properties:

- **Correctness:** *For every circuit $C \in \mathcal{C}$ with input length ℓ, $x \in \{0,1\}^\ell$, for every security parameter $\lambda \in \mathbb{N}$, it should hold that:*

$$\Pr\left[\begin{array}{c} \mathsf{gcsk} \leftarrow \mathsf{Gen}(1^\lambda), \\ ((k_1^0, k_1^1), \ldots, (k_\ell^0, k_\ell^1)) \leftarrow \mathsf{Grbl}(\mathsf{gcsk}), \\ \widehat{C} \leftarrow \mathsf{GrbC}(\mathsf{gcsk}, C) \\ C(x) \leftarrow \mathsf{EvalGC}(\widehat{C}, (k_1^{x_1}, \ldots, k_\ell^{x_\ell})) \end{array}\right] \ge 1 - \mathsf{negl}(\lambda)$$

- **Security:** *There exists a PPT simulator* SimGC *such that the following holds for every circuit $C \in \mathcal{C}$ of input length ℓ, $x \in \{0,1\}^\ell$,*

$$\left(\widehat{C}, k_1^{x_1}, \ldots, k_\ell^{x_\ell}\right) \cong_c \mathsf{SimGC}(1^\lambda, \phi(C), C(x)),$$

where:

  - $\mathsf{gcsk} \leftarrow \mathsf{Gen}(1^\lambda)$
  - $((k_1^0, k_1^1), \ldots, (k_\ell^0, k_\ell^1)) \leftarrow \mathsf{Grbl}(\mathsf{gcsk})$
  - $\widehat{C} \leftarrow \mathsf{GrbC}(\mathsf{gcsk}, C)$
  - $\phi(C)$ *is the topology of C.*

Theorem 17 ([Yao86, LP09, BHR12])**.** *Assuming the existence of one-way functions, there exists a secure garbling scheme* GC*.*