Two-Round Secure MPC from Indistinguishability Obfuscation*

Sanjam Garg[1], Craig Gentry[1], Shai Halevi[1], and Mariana Raykova[2]

[1] IBM T. J. Watson
[2] SRI International

Abstract. One fundamental complexity measure of an MPC protocol is its *round complexity*. Asharov et al. recently constructed the first three-round protocol for general MPC in the CRS model. Here, we show how to achieve this result with only two rounds. We obtain UC security with abort against static malicious adversaries, and fairness if there is an honest majority. Additionally the communication in our protocol is only proportional to the input and output size of the function being evaluated and independent of its circuit size. Our main tool is indistinguishability obfuscation, for which a candidate construction was recently proposed by Garg et al.

The technical tools that we develop in this work also imply virtual black box obfuscation of a new primitive that we call a *dynamic point function*. This primitive may be of independent interest.

1 Introduction

Secure multiparty computation (MPC) allows a group of mutually distrusting parties to jointly compute a function of their inputs without revealing their inputs to each other. This fundamental notion was introduced in the seminal works of [Yao+82, GMW+87], who showed that *any* function can be computed securely, even in the presence of malicious parties, provided the fraction of malicious parties is not too high. Since these fundamental feasibility results, much of the work related to MPC has been devoted to improving *efficiency*. There are various ways of measuring the efficiency of a MPC protocol, the most obvious being its computational complexity. In this paper, we focus on minimizing the *communication complexity* of MPC, primarily in terms of the number of *rounds* of interaction needed to complete the MPC protocol, but also in terms of the number of *bits* transmitted between the parties.

* The second and third authors were supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center (DoI/NBC) contract number D11PC20202. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

Y. Lindell (Ed.): TCC 2014, LNCS 8349, pp. 74–94, 2014.

1.1 Our Main Result: Two-Round MPC from Indistinguishability Obfuscation

Our main result is a *compiler* that transforms *any* MPC protocol into a *2-round* protocol in the CRS model. Our compiler is conceptually very simple, and it uses as its main tool *indistinguishability obfuscation* ($i\mathcal{O}$) [BGI+12]. Roughly, in the first round the parties commit to their inputs and randomness, and in the second round each party provides an *obfuscation* of their "next-message" function in the underlying MPC protocol. The parties then separately evaluate the obfuscated next-message functions to obtain the output.

A bit more precisely, our main result is as follows:

Informal Theorem. *Assuming indistinguishability obfuscation, CCA-secure public-key encryption, and statistically-sound noninteractive zero-knowledge, any multiparty function can be computed securely in just two rounds of broadcast.*

We prove that our MPC protocol resists static malicious corruptions in the UC setting [Can+01]. Moreover, the same protocol also achieves fairness if the set of corrupted players is a strict minority. Finally the communication in our protocol can be made to be only proportional to the input and output size of the function being evaluated and independent of its circuit size.

Minimizing round complexity is not just of theoretical interest. Low-interaction secure computation protocols are also applicable in the setting of computing on the web [HLP+11], where a single server coordinates the computation, and parties "log in" at different times without coordination.

1.2 Indistinguishability Obfuscation

Obfuscation was first rigorously defined and studied by Barak et al. [BGI+12]. Most famously, they defined a notion of *virtual black box (VBB)* obfuscation, and proved that this notion is impossible to realize in general – i.e., some functions are VBB unobfuscatable.

Barak et al. also defined a weaker notion of *indistinguishability obfuscation ($i\mathcal{O}$)*, which avoids their impossibility results. $i\mathcal{O}$ provides the same functionality guarantees as VBB obfuscation, but a weaker security guarantee. Namely, that for any two circuits C_0, C_1 of similar size that *compute the same function*, it is hard to distinguish an obfuscation of C_0 from an obfuscation of C_1. Barak et al. showed that $i\mathcal{O}$ is *always* realizable, albeit inefficiently: the $i\mathcal{O}$ can simply *canonicalize* the input circuit C by outputting the lexicographically first circuit that computes the same function. More recently, Garg et al. [GGH+13b] proposed an efficient construction of $i\mathcal{O}$ for all circuits, basing security in part on assumptions related to multilinear maps [GGH+13a].

It is clear that $i\mathcal{O}$ is a weaker primitive than VBB obfuscation. In fact, it is not hard to see that we cannot even hope to prove that $i\mathcal{O}$ implies one-way functions: Indeed, if $P = NP$ then one-way functions do not exist but $i\mathcal{O}$ does exist (since the canonicalizing $i\mathcal{O}$ from above can be implemented efficiently). Therefore we do not expect to build many "cryptographically interesting" tools just from $i\mathcal{O}$,

but usually need to combine it with other assumptions. (One exception is witness encryption [GGSW+13], which can be constructed from $i\mathcal{O}$ alone.)

It is known that $i\mathcal{O}$ can be combined with one-way functions (OWFs) to construct many powerful primitives such as public-key encryption, identity-based encryption, attribute-based encryption (via witness encryption), as well as NIZKs, CCA encryption, and deniable encryption [SW+12]. However, there are still basic tools that are trivially constructible from VBB obfuscation that we do not know how to construct from $i\mathcal{O}$ and OWFs: for example, collision-resistant hash functions, or compact homomorphic encryption. (Compact homomorphic encryption implies collision-resistant hash functions [IKO+05].) The main challenge in constructing primitives from $i\mathcal{O}$ is that the indistinguishability guarantee holds only in a limited setting: when the two circuits in question are perfectly functionally equivalent.

1.3 Our Techniques

To gain intuition and avoid technical complications, let us begin by considering how we would construct a 2-round protocol if we could use "perfect" VBB obfuscation. For starters, even with VBB obfuscation we still need at least two rounds of interaction, since a 1-round protocol would inherently allow the corrupted parties to repeatedly evaluate the "residual function" associated to the inputs of the honest parties on many different inputs of their choice (e.g., see [HLP+11]).

It thus seems natural to split our 2-round protocol into a commitment round in which all players "fix their inputs," and then an evaluation round where the output is computed. Moreover, it seems natural to use CCA-secure encryption to commit to the inputs and randomness, as this would enable a simulator to extract these values from the corrupted players.

As mentioned above, our idea for the second round is a simple compiler: take any (possibly highly interactive) underlying MPC protocol, and have each party obfuscate their "next-message" function in that protocol, one obfuscation for each round, so that the parties can independently evaluate the obfuscations to obtain the output. Party i's next-message function for round j in the underlying MPC protocol depends on its input x_i and randomness r_i (which are hardcoded in the obfuscations), it takes as input the transcript through round $j-1$, and it produces as output the next broadcast message.

However, there is a complication: unlike the initial interactive protocol, the obfuscations are susceptible to a "reset" attack – i.e., they can be evaluated on multiple inputs. To prevent such attacks, we ensure that the obfuscations can be used for evaluation only on a unique set of values – namely, values consistent with the inputs and randomness that the parties committed to in the first round, and the current transcript of the underlying MPC protocol. To ensure such consistency, naturally we use non-interactive zero-knowledge (NIZK) proofs. Since the NIZKs apply not only to the committed values of the first round, but also to the transcript as it develops in the second round, the obfuscations themselves must output these NIZKs "on the fly". In other words, the obfuscations are now

augmented to perform not only the next-message function, but also to prove that their output is consistent. Also, obfuscations in round j of the underlying MPC protocol verify NIZKs associated to obfuscations in previous rounds before providing any output.

If we used VBB obfuscation, we could argue security intuitively as follows. Imagine an augmented version of the underlying MPC protocol, where we prepend a round of commitment to the inputs and randomness, after which the parties (interactively) follow the underlying MPC protocol, except that they provide NIZK proofs that their messages are consistent with their committed inputs and randomness and the developing transcript. It is fairly easy to see that the security of this augmented protocol (with some minor modifications to how the randomness is handled) reduces to the security of the underlying MPC protocol (and the security of the CCA encryption and NIZK proof system). Now, remove the interaction by providing VBB obfuscations of the parties in the second round. These VBB obfuscations "virtually emulate" the parties of the augmented protocol while providing no additional information – in particular, the obfuscations output $\perp$ unless the input conforms exactly to the transcript of the underlying MPC protocol on the committed inputs and randomness; the obfuscations might accept many valid proofs, but since the proofs are statistically sound this gives no more information than one obtains in the augmented protocol.

Instead, we use indistinguishability obfuscation, and while the our protocol is essentially as described above, the proof of security is more subtle. Here, we again make use of the fact that the transcript in the underlying MPC protocol is completely determined by the commitment round, but in a different way. Specifically, there is a step in the proof where we change the obfuscations, so that instead of actually computing the next-message function (with proofs), these values are extracted and simply hardcoded in the obfuscations as the output on any accepting input. We show that these two types of obfuscations are functionally equivalent, and invoke $i\mathcal{O}$ to prove that they are indistinguishable. Once these messages have been "hardcoded" and separated from the computation, we complete the security proof using standard tricks. The most interesting remaining step in the proof is where we replace hardcoded real values with hardcoded simulated values generated by the simulator of the underlying MPC protocol.

1.4 Additional Results

Two-Round MPC with Low Communication. In our basic 2-round MPC protocol, the communication complexity grows polynomially with the circuit size of the function being computed. In Section 3.2, we show how to combine our basic 2-round protocol with *multikey fully homomorphic encryption* [LATV$^+$12] to obtain an MPC that is still only two rounds, but whose communication is basically independent of the circuit size. Roughly speaking, this protocol has a first round where the players encrypt their inputs and evaluate the function under a shared FHE key (and commit to certain values as in our basic protocol), followed by a second round where the players apply the second round of our basic protocol to decrypt the final FHE ciphertext.

Dynamic Point Functions. As a side effect of our technical treatment, we observe that $i\mathcal{O}$ can be used to extend the reach of (some) known VBB obfuscators. For example, we can VBB obfuscate *dynamic point functions*. In this setting, the obfuscation process is partitioned between two parties, the "point owner" Penny and the "function owner" Frank. Penny has a secret string (point) $x \in \{0,1\}^*$, and she publishes a commitment to her point $c_x = \mathrm{com}(x)$. Frank has a function $f : \{0,1\}^* \to \{0,1\}^*$ and knows c_x but not x itself. Frank wants to allow anyone who happens to know x to compute $f(x)$. A dynamic point function obfuscator allows Frank to publish an obfuscated version of the point function

$$F_{f,x}(z) = \begin{cases} f(x) & \text{if } z = x \\ \perp & \text{otherwise.} \end{cases}$$

The security requirement here is that $F_{f,x}$ is obfuscated in the strong VBB sense (and that c_x hides x computationally). We believe that this notion of dynamic point functions is interesting on its own and that it may find future applications.

1.5 Other Related Work

The round complexity of MPC has been studied extensively: both lower and upper bounds, for both the two-party and multiparty cases, in both the semi-honest and malicious settings, in plain, CRS and PKI models. See [AJLA+12, Section 1.3] for a thorough overview of this work.

Here, we specifically highlight the recent work of Asharov et al. [AJLA+12], which achieves 3-round MPC in the CRS model (and 2-round MPC in the PKI model) against static malicious adversaries. They use fully homomorphic encryption (FHE) [RAD+78, Gen+09], but not as a black box. Rather, they construct threshold versions of *particular* FHE schemes – namely, schemes by Brakerski, Gentry and Vaikuntanathan [BV+11, BGV+12] based on the learning with errors (LWE) assumption. (We note that Myers, Sergi and shelat [MSS+11] previously thresholdized a different FHE scheme based on the approximate gcd assumption [vDGHV+10], but their protocol required more rounds.)

In more detail, Asharov et al. observe that these particular LWE-based FHE schemes have a key homomorphic property. Thus, in the first round of their protocol, each party can encrypt its message under its own FHE key, and then the parties can use the key homomorphism to obtain encryptions of the inputs under a shared FHE key. Also, in the last round of their protocol, decryption is a simple one-round process, where decryption of the final ciphertext under the individual keys reveals the decryption under the shared key. In between, the parties use FHE evaluation to compute the encrypted output under the shared key. Unfortunately, they need a third (middle) round for technical reasons: LWE-based FHE schemes typically also have an "evaluation key" – namely, an encryption of a function of the secret key under the public key. They need the extra round to obtain an evaluation key associated to their shared key.

Recently, Gentry, Sahai and Waters [GSW+13] proposed an LWE-based FHE scheme without such an evaluation key. Unfortunately, eliminating the evalua-

tion key in their scheme does not seem to give 2-round MPC based on threshold FHE, since their scheme lacks the key homomorphism property needed by Asharov et al.

We note that our basic two-round protocol does not rely on any *particular* constructions for $i\mathcal{O}$ (or CCA-secure PKE or NIZK proofs), but rather uses these components as black boxes.

Our low-communication two-round protocol uses multikey FHE, but only as a black box. This protocol can be seen as a realization of what Asharov et al. were trying to achieve: a first round where the players encrypt their inputs and evaluate the function under a shared FHE key, followed by a second round where the players decrypt the final FHE ciphertext.

2 Preliminaries

In this section we will start by briefly recalling the definition of different notions essential for our study. We refer the reader to the full version of the paper [GGHR$^+$13] for additional background. The natural security parameter is λ, and all other quantities are implicitly assumed to be functions of λ. We use standard big-O notation to classify the growth of functions. We let $\mathsf{poly}(\lambda)$ denote an unspecified function $f(\lambda) = O(\lambda^c)$ for some constant c. A *negligible* function, denoted generically by $\mathsf{negl}(\lambda)$, is an $f(\lambda)$ such that $f(\lambda) = o(\lambda^{-c})$ for every fixed constant c. We say that a function is *overwhelming* if it is $1 - \mathsf{negl}(\lambda)$.

2.1 Indistinguishability Obfuscators

We will start by recalling the notion of indistinguishability obfuscation ($i\mathcal{O}$) recently realized in [GGH$^+$13b] using candidate multilinear maps[GGH$^+$13a].

Definition 1 (Indistinguishability Obfuscator ($i\mathcal{O}$)). *A uniform PPT machine $i\mathcal{O}$ is called an* indistinguishability obfuscator *for a circuit class $\{\mathcal{C}_\lambda\}$ if the following conditions are satisfied:*

- *For all security parameters $\lambda \in \mathbb{N}$, for all $C \in \mathcal{C}_\lambda$, for all inputs x, we have that*
$$\Pr[C'(x) = C(x) : C' \leftarrow i\mathcal{O}(\lambda, C)] = 1$$
- *For any (not necessarily uniform) PPT distinguisher D, there exists a negligible function α such that the following holds: For all security parameters $\lambda \in \mathbb{N}$, for all pairs of circuits $C_0, C_1 \in \mathcal{C}_\lambda$, we have that if $C_0(x) = C_1(x)$ for all inputs x, then*
$$\left| \Pr\left[D(i\mathcal{O}(\lambda, C_0)) = 1\right] - \Pr\left[D(i\mathcal{O}(\lambda, C_1)) = 1\right] \right| \leq \alpha(\lambda)$$

Definition 2 (Indistinguishability Obfuscator for NC^1).
A uniform PPT machine $i\mathcal{O}$ is called an indistinguishability obfuscator *for NC^1 if for all constants $c \in \mathbb{N}$, the following holds: Let $\mathcal{C}_\lambda$ be the class of circuits of depth at most $c \log \lambda$ and size at most λ. Then $i\mathcal{O}(c, \cdot, \cdot)$ is an indistinguishability obfuscator for the class $\{\mathcal{C}_\lambda\}$.*

Definition 3 (Indistinguishability Obfuscator for $P/poly$). *A uniform PPT machine $i\mathcal{O}$ is called an* indistinguishability obfuscator *for $P/poly$ if the following holds: Let $\mathcal{C}_\lambda$ be the class of circuits of size at most λ. Then $i\mathcal{O}$ is an indistinguishability obfuscator for the class $\{\mathcal{C}_\lambda\}$.*

2.2 Semi-honest MPC

We will also use a semi-honest n-party computation protocol π for any functionality f in the stand-alone setting. The existence of such a protocol follows from the existence of semi-honest 1-out-of-2 oblivious transfer [Yao$^+$82, GMW$^+$87] protocols. Now we build some notation that we will use in our construction.

Let $\mathcal{P} = \{P_1, P_2, \ldots P_n\}$ be the set of parties participating in a t round protocol π. Without loss of generality, in order to simplify notation, we will assume that in each round of π, each party broadcasts a single message that depends on its input and randomness and on the messages that it received from all parties in all previous rounds. (We note that we can assume this form without loss of generality, since in our setting we have broadcast channels and CCA-secure encryption, and we only consider security against static corruptions.) We let $m_{i,j}$ denote the message sent by the i^{th} party in the j^{th} round. We define the function π_i such that $m_{i,j} = \pi_i(x_i, r_i, M_{j-1})$ where $m_{i,j}$ is the j^{th} message generated by party P_i in protocol π with input x_i, randomness r_i and the series of previous messages M_{j-1}

$$M_{j-1} = \begin{pmatrix} m_{1,1} & m_{2,1} & \ldots & m_{n,1} \\ m_{1,2} & m_{2,2} & \ldots & m_{n,2} \\ \vdots & \ddots & & \\ m_{1,j-1} & m_{2,j-1} & \ldots & m_{n,j-1} \end{pmatrix}$$

sent by all parties in π.

3 Our Protocol

In this section, we provide our construction of a two-round MPC protocol.

Protocol Π. We start by giving an intuitive description of the protocol. A formal description appears in Figure 1. The basic idea of our protocol is to start with an arbitrary round semi-honest protocol π and "squish" it into a two round protocol using indistinguishability obfuscation. The first round of our protocol helps set the stage for the "virtual" execution of π via obfuscations that all the parties provide in the second round.

The common reference string in our construction consists of a CRS σ for a NIZK Proof system and a public key pk corresponding to a CCA-secure public key encryption scheme. Next, the protocol proceeds in two rounds as follows:

Protocol Π

Protocol Π uses an Indistinguishability Obfuscator $i\mathcal{O}$, a NIZK proof system (K, P, V), a CCA-secure PKE scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with perfect correctness and an n-party semi-honest MPC protocol π.

Private Inputs: Party P_i for $i \in [n]$, receives its input x_i.

Common Reference String: Let $\sigma \leftarrow K(1^\lambda)$ and $(pk, \cdot) \leftarrow \mathsf{Gen}(1^\lambda)$ and then output (σ, pk) as the common reference string.

Round 1: Each party P_i proceeds as:

- $c_i = \mathsf{Enc}(i||x_i)$ and,
- $\forall j \in [n]$, sample randomness $r_{i,j} \in \{0,1\}^\ell$ and generate $d_{i,j} = \mathsf{Enc}(i||r_{i,j})$. (Here ℓ is the length of the maximum number of random coins needed by any party in π.)

It then sends $Z_i = \{c_i, \{d_{i,j}\}_{j\in[n]}\}$ to every other party.

Round 2: P_i generates:

- For every $j \in [n]$, $j \neq i$ generate $\gamma_{i,j}$ as the NIZK proof under σ for the NP-statement:
$$\left\{\exists\ \rho_{r_{i,j}} \ \middle|\ d_{i,j} = \mathsf{Enc}(i||r_{i,j}; \rho_{r_{i,j}})\right\}. \tag{1}$$
- A sequence of obfuscations $(i\mathcal{O}_{i,1}, \ldots i\mathcal{O}_{i,t})$ where $i\mathcal{O}_{i,j}$ is the obfuscation of the program $\mathsf{Prog}_{i,j}^{0,x_i,\rho_{x_i},r_{i,i},\rho_{r_{i,i}},\{Z_i\},0^{\ell_{i,j}}}$. (Where $\ell_{i,j}$ is output length of the program $\mathsf{Prog}_{i,j}$.)
- It sends $(\{r_{i,j}, \gamma_{i,j}\}_{j\in[n],j\neq i}, \{i\mathcal{O}_{i,j}\}_{j\in[t]})$ to every other party.

Evaluation (MPC in the Head): For each $j \in [t]$ proceed as follows:

- For each $i \in [n]$, evaluate the obfuscation $i\mathcal{O}_{i,j}$ of program $\mathsf{Prog}_{i,j}$ on input $(R, \Gamma, M_{j-1}, \Phi_{j-1})$ where

$$R = \begin{pmatrix} \cdot & r_{2,1} & \ldots & r_{n,1} \\ r_{1,2} & \cdot & \ldots & r_{n,2} \\ \vdots & \ddots & & \\ r_{1,n} & r_{2,n} & \ldots & \cdot \end{pmatrix}, \ \Gamma = \begin{pmatrix} \cdot & \gamma_{2,1} & \ldots & \gamma_{n,1} \\ \gamma_{1,2} & \cdot & \ldots & \gamma_{n,2} \\ \vdots & \ddots & & \\ \gamma_{1,n} & \gamma_{2,n} & \ldots & \cdot \end{pmatrix}$$

$$M_{j-1} = \begin{pmatrix} m_{1,1} & m_{2,1} & \ldots & m_{n,1} \\ m_{1,2} & m_{2,2} & \ldots & m_{n,2} \\ \vdots & \ddots & & \\ m_{1,j-1} & m_{2,j-1} & \ldots & m_{n,j-1} \end{pmatrix}, \ \Phi = \begin{pmatrix} \phi_{1,1} & \phi_{2,1} & \ldots & \phi_{n,1} \\ \phi_{1,2} & \phi_{2,2} & \ldots & \phi_{n,2} \\ \vdots & \ddots & & \\ \phi_{1,j-1} & \phi_{2,j-1} & \ldots & \phi_{n,j-1} \end{pmatrix}$$

- And obtain, $m_{1,j}, \ldots, m_{n,j}$ and $\phi_{1,j}, \ldots, \phi_{n,j}$.

Finally each party P_i outputs $m_{i,t}$.

Fig. 1. Two Round MPC Protocol

$$\mathbf{Prog}_{i,j}^{\mathsf{flag},x_i,\rho_{x_i},r_{i,i},\rho_{r_{i,i}},\{Z_i\},\mathsf{fixedOutput}}$$

Program $\mathbf{Prog}_{i,j}^{\mathsf{flag},x_i,\rho_{x_i},r_{i,i},\rho_{r_{i,i}},\{Z_i\},\mathsf{fixedOutput}}$ takes as input $(R, \Gamma, M_{j-1}, \Phi)$ as defined above and outputs $m_{i,j}$ and $\phi_{i,j}$. Specifically, it proceeds as follows:

- $\forall p, q \in [n]$ such that $p \neq q$ check that $\gamma_{p,q}$ is an accepting proof under σ for the NP-statement:

$$\left\{\exists\ \rho_{r_{p,q}}\ \mid\ d_{p,q} = \mathsf{Enc}(p||r_{p,q}; \rho_{r_{p,q}})\right\}$$

- $\forall p \in [n], q \in [j-1]$ check that $\phi_{p,q}$ is an accepting proof for the NP-statement

$$\left\{\begin{array}{l}\exists\ (x_p, r_{p,p}, \rho_{x_p}, \rho_{r_{p,p}})\ \mid \\ \left(c_p = \mathsf{Enc}(p||x_p; \rho_{x_p}) \bigwedge d_{p,p} = \mathsf{Enc}(p||r_{p,p}, \rho_{r_{p,p}}) \bigwedge m_{p,q} = \pi_p(x_p, \oplus_{k\in[n]} r_{k,p}, M_{q-1})\right)\end{array}\right\}$$

- If the checks above fail, output $\perp$. Otherwise, if $\mathsf{flag} = 0$ then output $(\pi_i(x_i, \oplus_{j\in[n]} r_{j,i}, M_{j-1}), \phi_{i,j})$ where $\phi_{i,j}$ is the proof for the NP-statement: (under some fixed randomness)

$$\left\{\begin{array}{l}\exists\ (x_i, r_{i,i}, \rho_{x_i}, \rho_{r_{i,i}})\ \mid \\ \left(c_i = \mathsf{Enc}(i||x_i; \rho_{x_i})\ \bigwedge\ d_{i,i} = \mathsf{Enc}(i||r_{i,i}, \rho_{r_{i,i}})\ \bigwedge\ m_{i,j} = \pi_i(x_i, \oplus_{j\in[n]} r_{j,i}, M_{j-1})\right)\end{array}\right\}$$

Otherwise, output $\mathsf{fixedOutput}$.

Fig. 2. Obfuscated Programs in the Protocol

Round 1: In the first round, the parties "commit" to their inputs and randomness, where the commitments are generated using the CCA-secure encryption scheme. The committed randomness will be used for coin-flipping and thereby obtaining unbiased random coins for all parties. Specifically, every party P_i, proceeds by encrypting its input x_i under the public key pk. Let c_i be the ciphertext. P_i also encrypts randomness $r_{i,j}$ for every $j \in [n]$. Let the ciphertext encrypting $r_{i,j}$ be denoted by $d_{i,j}$. Looking ahead the random coins P_i uses in the execution of π will be $s_i = \oplus_j r_{j,i}$. P_i broadcasts $\{c_i, \{d_{i,j}\}_j\}$ to everyone.

Round 2: In the second round parties will broadcast obfuscations corresponding to the next message function of π allowing for a "virtual emulation" of the interactive protocol π. Every party P_i proceeds as follows:

- P_i reveals the random values $\{r_{i,j}\}_{j\neq i\in[n]}$ and generates proofs $\{\gamma_{i,j}\}_{j\neq i\in[n]}$ that these are indeed the values that are encrypted in the ciphertexts $\{d_{i,j}\}_{j\neq i\in[n]}$.
- Recall that the underlying protocol π is a t round protocol where each party broadcasts one message per round. Each player P_i generates t obfuscations of its next-round function, $(i\mathcal{O}_{i,1}, \ldots, i\mathcal{O}_{i,t})$.
  In more detail, each $i\mathcal{O}_{i,k}$ is an obfuscation of a function $F_{i,k}$ that takes as input the $r_{i,j}$ values sent by all the parties along with the proofs that they are well-formed, and also all the π-messages that were broadcast upto round $k-1$, along with the proof of correct generation of these messages. (These proofs are all with respect to the ciphertexts generated in first round and the revealed $r_{i,j}$ values.) The output of the function

$F_{i,j}$ is the next message of P_i in π, along with a NIZK proof that it was generated correctly.

P_i broadcasts all the values $\{r_{i,j}\}_{j\neq i\in[n]}$, $\{\gamma_{i,j}\}_{j\neq i\in[n]}$, and $\{i\mathcal{O}_{i,k}\}_{k\in[t]}$.

Evaluation: After completion of the second round each party can independently "virtually" evaluate the protocol π using the obfuscations provided by each of the parties and obtain the output.

Theorem 1. *Let f be any deterministic poly-time function with n inputs and single output. Assume the existence of an Indistinguishability Obfuscator $i\mathcal{O}$, a NIZK proof system (K, P, V), a CCA secure PKE scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with perfect correctness and an n-party semi-honest MPC protocol π. Then the protocol Π presented in Figure 1 UC-securely realizes the ideal functionality $\mathcal{F}_f$ in the $\mathcal{F}_{CRS}$-hybrid model.*

3.1 Correctness and Proof of Security

Correctness. The correctness of our protocol Π in Figure 1 follows from the correctness of the underlying semi-honest MPC protocol and the other primitives used. Next we will argue that all the messages sent in the protocol Π are of polynomial length and can be computed in polynomial time. It is easy to see that all the messages of round 1 are polynomially long. Again it is easy to see that the round 2 messages besides the obfuscations themselves are of polynomial length.

We will now argue that each obfuscation sent in round 2 is also polynomially long. Consider the obfuscation $i\mathcal{O}_{i,j}$, which obfuscates $\mathsf{Prog}_{i,j}$; we need to argue that this program for every i, j is only polynomially long. Observe that this program takes as input $(R, \Gamma, M_{i-1}, \Phi_{j-1})$, where Γ and Φ_{j-1} consist of polynomially many NIZK proofs. This program roughly proceeds by first checking that all the proofs in Γ and Φ_{j-1} are accepting. If the proofs are accepting then Prog outputs $m_{i,j}$ and $\phi_{i,j}$.

Observe that Γ and Φ_{j-1} are proofs of NP-statements each of which is a fixed polynomial in the description of the next message function of the protocol π. Also observe that the time taken to evaluate $m_{i,j}$ and $\phi_{i,j}$ is bounded a fixed polynomial. This allows us to conclude that all the computation done by $\mathsf{Prog}_{i,j}$ can be bounded by a fixed polynomial.

Security. Let $\mathcal{A}$ be a malicious, static adversary that interacts with parties running the protocol Π from Figure 1 in the $\mathcal{F}_{CRS}$-hybrid model. We construct an ideal world adversary $\mathcal{S}$ with access to the ideal functionality $\mathcal{F}_f$, which simulates a real execution of Π with $\mathcal{A}$ such that no environment $\mathcal{Z}$ can distinguish the ideal world experiment with $\mathcal{S}$ and $\mathcal{F}_f$ from a real execution of Π with $\mathcal{A}$.

We now sketch the description of the simulator and the proof of security, restricting ourselves to the stand-alone setting. The fully detailed description of our simulator and the proof of indistinguishability are provided in Appendix A. Those more formal proofs are given for the general setting of UC-security.

Our simulator $\mathcal{S}$ roughly proceeds as follows:

- **Common reference string:** Recall that the common reference string in our construction consists of a CRS σ for a NIZK Proof system and a public key pk corresponding to a CCA secure public key encryption scheme. Our simulator uses the simulator of the NIZK proof system in order to generate the reference string σ. Note that the simulator for NIZK proof system also generates some trapdoor information that can be used to generate simulated NIZK proofs. Our simulator saves that for later use. $\mathcal{S}$ also generates the public key pk along with its secret key sk, which it will later use to decrypt ciphertexts generated by the adversary.
- **Round 1:** Recall that in round 1, honest parties generate ciphertexts corresponding to encryptions of their inputs and various random coins. Our simulator just generates encryptions of the zero-string on behalf of the honest parties. Also $\mathcal{S}$ uses the knowledge of the secret key sk to extract the input and randomness that the adversarial parties encrypt.
- **Round 2:** Recall that in the second round the honest parties are required to "open" some of the randomness values committed to in round 1 along with obfuscations necessary for execution of π.
  $\mathcal{S}$ proceeds by preparing a simulated transcript of the execution of π using the malicious party inputs previously extracted and the output obtained from the ideal functionality, which it needs to force onto the malicious parties. $\mathcal{S}$ opens the randomness on behalf of honest parties such that the randomness of malicious parties becomes consistent with the simulated transcript and generates simulated proofs for the same. The simulator generates the obfuscations on behalf of honest parties by hard-coding the messages as contained in the simulated transcript. The obfuscations also generate proofs proving that the output was generated correctly. Our simulator hard-codes these proofs in the obfuscations as well.

Very roughly, our proof proceeds by first changing all the obfuscations $\mathcal{S}$ generates on behalf of honest parties to output fixed values. The statistical soundness of the NIZK proof system allows us to base security on the weak notion of indistinguishability obfuscation. Once this change has been made, in a sequence of hybrids we change from honest execution of the underlying semi-honest MPC protocol to a the simulated execution. We refer the reader to Appendix A for a complete proof.

3.2 Extensions

Low Communication. Our protocol Π (as described in Figure 1) can be used to UC-securely realize any functionality $\mathcal{F}_f$. However the communication complexity of this protocol grows polynomially in the size of the circuit evaluating function f and the security parameter λ. We would like to remove this restriction and construct a protocol Π' whose communication complexity is independent of the the function being evaluated.

A key ingredient of our construction is *multikey fully homomorphic encryption* [LATV$^+$12]. Intuitively, multikey FHE allows us to evaluate any circuit on

Protocol Π'

Let Π be the MPC Protocol from Figure 1.
Let $(\mathsf{Setup}_{MK}, \mathsf{Encrypt}_{MK}, \mathsf{Eval}_{MK}, \mathsf{Decrypt}_{MK})$ be a multikey FHE scheme.
Private Inputs: Party P_i for $i \in [n]$, receives its input x_i.
Common Reference String: Generate the CRS corresponding to Π.

Round 1: P_i proceeds as follows:
- $(pk_i, sk_i) \leftarrow \mathsf{Setup}_{MK}(1^\lambda; \rho_i)$ and generates encryption $c_i := \mathsf{Encrypt}_{MK}(pk_i, x_i; \varrho_i)$.
- Generates the first round message Z_i of Π playing as P_i with input (x_i, ρ_i, ϱ_i). (Recall that the first message of Π does not depend on the function Π is used to evaluate.)
- Sends3 (pk_i, c_i, Z_i) to all parties.

Round 2: Every party P_i computes $c^* := \mathsf{Eval}_{MK}(C, (c_1, pk_1), \ldots, (c_n, pk_n))$. P_i generates P_i's second round message of Π, where Π computes the following function:
- For every $i \in [n]$, check if $(pk_i, sk_i) \leftarrow \mathsf{Setup}_{MK}(1^\lambda; \rho_i)$ and $c_i := \mathsf{Encrypt}_{MK}(pk_i, x_i; \varrho_i)$.
- If all the checks pass then output $\mathsf{Decrypt}_{MK}(sk_1, \ldots, sk_n, c^*)$ and otherwise output $\perp$.

Evaluation: P_i outputs the output of P_i in Π.

Fig. 3. Two Round MPC Protocol with Low Communication Complexity

ciphertexts that might be encrypted under different public keys. To guarantee semantic security, decryption requires all of the corresponding secret keys. We refer the reader to the full version of the paper [GGHR+13] for more details.

Our protocol Π' works by invoking Π. Recall that Π proceeds in two rounds. Roughly speaking, in the first stage parties commit to their inputs, and in the second round the parties generate obfuscations that allow for "virtual" execution of sub-protocol π on the inputs committed in the first round. Our key observation here is that the function that the sub-protocol π evaluates does not have to be specified until the second round.

We will now give a sketch of our protocol Π'. Every party P_i generates a public key pk_i and a secret key sk_i using the setup algorithm of the multikey FHE scheme. It then encrypts its input x_i under the public key pk_i and obtains ciphertext c_i. It then sends (pk_i, c_i) to everyone along with the first message of Π with input the randomness used in generation of pk_i and c_i. This completes the first round. At this point, all parties can use the values $((pk_1, c_1), \ldots, (pk_n, c_n))$ to obtain an encryption of $f(x_1, \ldots x_n)$, where f is the function that we want to compute. The second round of protocol Π can be used to decrypt this value. A formal description of the protocol appears in Figure 3.

Theorem 2. *Under the same assumptions as in Theorem 1 and assuming the semantic security of the multikey FHE scheme, the protocol Π' presented in Figure 3 UC-securely realizes the ideal functionality $\mathcal{F}_f$ in the $\mathcal{F}_{CRS}$-hybrid model.*

Furthermore the communication complexity of protocol Π' is polynomial in the input lengths of all parties and the security parameter. (It is independent of the size of f.)

Proof. The correctness of the our protocol Π' follows from the correctness of the protocol Π and the correctness of the multikey FHE scheme. Observe that the compactness of the multikey FHE implies that the ciphertext c^* evaluated in Round 2 on the description of Protocol Π (Figure 3) is independent of the size of the function f being evaluated. Also note that no other messages in the protocol depend on the function f. This allows us to conclude that the communication complexity of protocol Π' is independent of the size of f.

We defer the formal description of our simulator and the proof of indistinguishability to the full version of the paper [GGHR$^+$13]. ∎

General Functionality. Our basic MPC protocol as described in Figure 1 only considers deterministic functionalities (See [GGHR$^+$13]) where all the parties receive the same output. We would like to generalize it to handle randomized functionalities and individual outputs (just as in [GGHR$^+$13, AJW$^+$11]). First, the standard transformation from a randomized functionality to a deterministic one (See [Gol$^+$04, Section 7.3]) works for this case as well. In this transformation, instead of computing some randomized function $g(x_1, \ldots x_n; r)$, the parties compute the deterministic function $f((r_1, x_1), \ldots, (r_n, x_n)) \stackrel{def}{=} g(x_1, \ldots, x_n; \oplus_{i=1}^{n} r_i)$. We note that this computation does not add any additional rounds.

Next, we move to individual outputs. Again, we use a standard transformation (See [LP$^+$09], for example). Given a function $g(x_1, \ldots, x_n) \to (y_1, \ldots, y_n)$, the parties can evaluate the following function which has a single output:

$$f((k_1, x_1), \ldots, (k_n; x_n)) = (g_1(x_1, \ldots, x_n) \oplus k_1 || \ldots || g_n(x_1, \ldots, x_n) \oplus k_n)$$

where $a||b$ denotes a concatenation of a with b, g_i indicates the i^{th} output of g, and k_i is randomly chosen by the i^{th} party. Then, the parties can evaluate f, which is a single output functionality, instead of g. Subsequently every party P_i uses its secret input k_i to recover its own output. The only difference is that f has one additional exclusive-or gate for every circuit-output wire. Again, this transformation does not add any additional rounds of interaction.

Corollary 1. *Let f be any (possibly randomized) poly-time function with n inputs and n outputs. Assume the existence of an Indistinguishability Obfuscator $i\mathcal{O}$, a NIZK proof system (K, P, V), a CCA secure PKE scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with perfect correctness and an n-party semi-honest MPC protocol π. Then the protocol Π presented in Figure 1 UC-securely realizes the ideal functionality $\mathcal{F}_f$ in the $\mathcal{F}_{CRS}$-hybrid model.*

Common Random String vs Common Reference String. Our basic MPC protocol as described in Figure 1 uses a common reference string. We can adapt the construction to work in the setting of common random string by assuming the

existence of a CCA secure public-key encryption scheme with perfect correctness and pseudorandom public keys and a NIZK scheme [FLS+90]. See [GGHR+13] for details.

Fairness. We note that the same protocol Π can be used to securely and fairly UC-realize the generalized functionality in the setting of honest majority, by using a fair semi-honest MPC protocol for π.

4 Applications

In this section we will discuss additional applications of our results.

4.1 Secure Computation on the Web

In a recent work, Halevi, Lindell and Pinkas [HLP+11] studied secure computation in a client-server model where each client connects to the server once and interacts with it, without any other client necessarily being connected at the same time. They show that, in such a setting, only limited security is achievable. However, among other results, they also point out that if we can get each of the players to connect twice to the server (rather than once), then their protocols can be used for achieving the standard notion of privacy.

One key aspect of the two-pass protocols of Halevi et. al [HLP+11] is that there is a preset order in which the clients must connect to the server. Our protocol Π from Section 3 directly improves on the results in this setting by achieving the same two-pass protocol, but without such a preset order. Also, we achieve this result in the common reference/random string model, while the original protocols of Halevi et. al [HLP+11] required a public key setup.

4.2 Black-Box Obfuscation for More Functions

In this subsection, we generalize the class of circuits that can be obfuscated according to the strong (virtual black box (VBB) notion of obfuscation. This application does not build directly on our protocol for two-round MPC. Rather, the main ideas here are related to ideas (particularly within the security proof) that arose in our MPC construction.

Our Result. Let $\mathcal{C}$ be a class of circuits that we believe to be VBB obfuscatable, e.g., point functions or conjunctions. Roughly speaking, assuming indistinguishability obfuscation, we show that a circuit C can be VBB obfuscated if there exists a circuit C' such that $C' \in \mathcal{C}$ and $C(x) = C'(x)$ for every input x. The non-triviality of the result lies in the fact that it might not be possible to efficiently recover C' from C. We refer the reader to the full version of the paper [GGHR+13] for a formal statement and proof.

Dynamic Point Function Obfuscation. We will now highlight the relevance of the results presented above with an example related to point functions. We know how to VBB obfuscate point functions. Now, consider a setting of three players. Player 1 generates a (perfectly binding) commitment to a value x. Player 2 would like to generate an obfuscation of an arbitrary function f that allows an arbitrary Player 3, if he knows x, to evaluate f on input x alone (and nothing other than x). Our construction above enables such obfuscation. We stress that the challenge here is that Player 2 is not aware of the value x, which is in fact computationally hidden from it.

References

[AJLA+12] Asharov, G., Jain, A., López-Alt, A., Tromer, E., Vaikuntanathan, V., Wichs, D.: Multiparty computation with low communication, computation and interaction via threshold FHE. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 483–501. Springer, Heidelberg (2012)

[AJW+11] Asharov, G., Jain, A., López-Alt, A., Tromer, E., Vaikuntanathan, V., Wichs, D.: Multiparty computation with low communication, computation and interaction via threshold FHE. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 483–501. Springer, Heidelberg (2012)

[BGI+12] Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. J. ACM 59(2), 6 (2012)

[BGV+12] Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. In: ITCS (2012)

[BV+11] Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) lwe. In: FOCS, pp. 97–106 (2011)

[Can+01] Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd Annual Symposium on Foundations of Computer Science, Las Vegas, Nevada, USA, October 14-17, pp. 136–145. IEEE Computer Society Press (2001)

[FLS+90] Feige, U., Lapidot, D., Shamir, A.: Multiple non-interactive zero knowledge proofs based on a single random string. In: Proceedings of the 31st Annual Symposium on Foundations of Computer Science, vol. 1, pp. 308–317 (1990)

[Gen+09] Gentry, C.: A fully homomorphic encryption scheme. PhD thesis, Stanford University (2009), `http://crypto.stanford.edu/craig`

[GGH+13a] Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 75–92. Springer, Heidelberg (2013)

[GGH+13b] Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: FOCS 2013. IEEE (to appear, 2013), `http://eprint.iacr.org/2013/451`

[GGHR+13] Garg, S., Gentry, C., Halevi, S., Raykova, M.: Two-round secure mpc from indistinguishability obfuscation. Cryptology ePrint Archive, Report 2013/601 (2013), http://eprint.iacr.org/

[GGSW+13] Garg, S., Gentry, C., Sahai, A., Waters, B.: Witness encryption and its applications. In: STOC (2013)

[GMW+87] Garg, S., Gentry, C., Sahai, A., Waters, B.: Witness encryption and its applications. In: STOC (2013)

[Gol+04] Goldreich, O.: Foundations of Cryptography: Basic Applications, vol. 2. Cambridge University Press, Cambridge (2004)

[GSW+13] Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 75–92. Springer, Heidelberg (2013)

[HLP+11] Halevi, S., Lindell, Y., Pinkas, B.: Secure computation on the web: Computing without simultaneous interaction. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 132–150. Springer, Heidelberg (2011)

[IKO+05] Ishai, Y., Kushilevitz, E., Ostrovsky, R.: Sufficient conditions for collision-resistant hashing. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 445–456. Springer, Heidelberg (2005)

[LATV+12] López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: STOC, pp. 1219–1234 (2012)

[LP+09] Lindell, Y., Pinkas, B.: A proof of security of Yao's protocol for two-party computation. Journal of Cryptology 22(2), 161–188 (2009)

[MSS+11] Myers, S., Sergi, M., Shelat, A.: Threshold fully homomorphic encryption and secure computation. IACR Cryptology ePrint Archive 2011, 454 (2011)

[RAD+78] Rivest, R., Adleman, L., Dertouzos, M.L.: On data banks and privacy homomorphisms. In: Foundations of Secure Computation, pp. 169–180 (1978)

[SW+12] Sahai, A., Waters, B.: How to use indistinguishability obfuscation: Deniable encryption, and more. IACR Cryptology ePrint Archive 2013, 454 (2013)

[vDGHV+10] van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010)

[Yao+82] Yao, A.C.: Protocols for secure computations. In: 23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, November 3-5, pp. 160–164. IEEE Computer Society Press (1982)

A Proof of Security of Theorem 1

Let $\mathcal{A}$ be a malicious, static adversary that interacts with parties running the protocol Π from Figure 1 in the $\mathcal{F}_{CRS}$-hybrid model. We construct an ideal world adversary $\mathcal{S}$ with access to the ideal functionality $\mathcal{F}_f$, which simulates a real execution of Π with $\mathcal{A}$ such that no environment $\mathcal{Z}$ can distinguish the ideal world experiment with $\mathcal{S}$ and $\mathcal{F}_f$ from a real execution of Π with $\mathcal{A}$.

Recall that $\mathcal{S}$ interacts with the ideal functionality $\mathcal{F}_f$ and with the environment $\mathcal{Z}$. The ideal adversary $\mathcal{S}$ starts by invoking a copy of $\mathcal{A}$ and running a

simulated interaction of $\mathcal{A}$ with the environment $\mathcal{Z}$ and the parties running the protocol. Our simulator $\mathcal{S}$ proceeds as follows:

Simulated CRS: The common reference string is chosen by $\mathcal{S}$ in the following manner (recall that $\mathcal{S}$ chooses the CRS for the simulated $\mathcal{A}$ as we are in the $\mathcal{F}_{CRS}$-hybrid model):

1. $\mathcal{S}$ generates $(\sigma, \tau) \leftarrow S_1(1^\lambda)$, the simulated common reference string for the NIZK proof system (K, P, V) with simulator $S = (S_1, S_2)$.
2. $\mathcal{S}$ runs the setup algorithm $\mathsf{Gen}(1^\lambda)$ of the CCA secure encryption scheme and obtains a public key pk and a secret key sk.

$\mathcal{S}$ sets the common reference string to equal (σ, pk) and locally stores (τ, sk). (The secret key sk will be later used to extract inputs of the corrupted parties and the trapdoor τ for the simulated CRS σ will be used to generate simulated proofs.)

Simulating the Communication with $\mathcal{Z}$: Every input value that $\mathcal{S}$ receives from $\mathcal{Z}$ is written on $\mathcal{A}$'s input tape. Similarly, every output value written by $\mathcal{A}$ on its own output tape is directly copied to the output tape of $\mathcal{S}$.

Simulating Actual Protocol Messages in Π: Note that there might be multiple sessions executing concurrently. Let sid be the session identifier for one specific session. We will specify the simulation strategy corresponding to this specific session. The simulator strategy for all other sessions will be the same. Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be the set of parties participating in the execution of Π corresponding to the session identified by the session identifier sid. Also let $\mathcal{P}^{\mathcal{A}} \subseteq \mathcal{P}$ be the set of parties corrupted by the adversary $\mathcal{A}$. (Recall that we are in the setting with static corruption.)

In the subsequent exposition we will assume that at least one party is honest. If no party is honest then the simulator does not need to do anything else.

Round 1 Messages $\mathcal{S} \to \mathcal{A}$: In the first round $\mathcal{S}$ must generate messages on behalf of the honest parties, i.e. parties in the set $\mathcal{P} \backslash \mathcal{P}^{\mathcal{A}}$. For each party $P_i \in \mathcal{P} \backslash \mathcal{P}^{\mathcal{A}}$ our simulator proceeds as:

1. $c_i = \mathsf{Enc}(i||0^{\ell_{in}})$ and, (recall that ℓ_{in} is the length of inputs of all parties)
2. $\forall j \in [n]$, and generate $d_{i,j} = \mathsf{Enc}(i||0^{\ell})$. (Recall that ℓ is the length of the maximum number of random coins needed by any party in π.)

It then sends $Z_i = \{c_i, \{d_{i,j}\}_{j \in [n]}\}$ to $\mathcal{A}$ on behalf of party P_i.

Round 1 Messages $\mathcal{A} \to \mathcal{S}$: Also in the first round the adversary $\mathcal{A}$ generates the messages on behalf of corrupted parties in $\mathcal{P}^{\mathcal{A}}$. For each party $P_i \in \mathcal{P}^{\mathcal{A}}$ our simulator proceeds as:

1. Let $Z_i = \{c_i, \{d_{i,j}\}_{j \in [n]}\}$ be the message that $\mathcal{A}$ sends on behalf of P_i. Our simulator $\mathcal{S}$ decrypts the ciphertexts using the secret key sk. In particular $\mathcal{S}$

sets $x'_i = \mathsf{Dec}(sk, c_i)$ and $r'_{i,j} = \mathsf{Dec}(sk, d_{i,j})$. Obtain $x_i \in \{0,1\}^{\ell_{in}}$ such that $x'_i = i||x_i$. If x'_i is not of this form the set $x_i = \perp$. Similarly obtain $r_{i,j}$ from $r'_{i,j}$ for every j setting the value to $\perp$ in case it is not of the right format.
2. $\mathcal{S}$ sends $(\mathsf{input}, \mathsf{sid}, \mathcal{P}, P_i, x_i)$ to $\mathcal{F}_f$ on behalf of the corrupted party P_i. It saves the values $\{r_{i,j}\}_j$ for later use.

Round 2 Messages $\mathcal{S} \to \mathcal{A}$: In the second round $\mathcal{S}$ must generate messages on behalf of the honest parties, i.e. parties in the set $\mathcal{P} \backslash \mathcal{P}^{\mathcal{A}}$. $\mathcal{S}$ proceeds as follows:

- $\mathcal{S}$ obtains the output $(\mathsf{output}, \mathsf{sid}, \mathcal{P}, y)$ from the ideal functionality $\mathcal{F}_f$ and now it needs to force this output onto the adversary $\mathcal{A}$.
- In order to force the output, the simulator $\mathcal{S}$ executes the simulator $\mathcal{S}_\pi$ and obtains a simulated transcript. The simulated transcript specifies the random coins of all the parties in $\mathcal{P}^{\mathcal{A}}$ and the protocol messages. Let s_i denote the random coins of party $P_i \in \mathcal{P}^{\mathcal{A}}$ and let $m_{i,j}$ for $i \in [n]$ and $j \in [t]$ denote the protocol messages. (Semi-honest security of protocol π implies the existence of such a simulator.)
- For each $P_j \in \mathcal{P}^{\mathcal{A}}$ sample $r_{i,j}$ randomly in $\{0,1\}^\ell$ for each $P_i \in \mathcal{P} \backslash \mathcal{P}^{\mathcal{A}}$ subject to the constraint that $\oplus_{i=1}^n r_{i,j} = s_j$.
- For each $P_i \in \mathcal{P} \backslash \mathcal{P}^{\mathcal{A}}$, $\mathcal{S}$ proceeds as follows:

1. For every $j \in [n]$, $j \neq i$ generate $\gamma_{i,j}$ as a simulated NIZK proof under σ for the NP-statement:
$$\left\{ \exists\ \rho_{r_{i,j}} \mid d_{i,j} = \mathsf{Enc}(i||r_{i,j}; \rho_{r_{i,j}}) \right\}.$$
2. A sequence of obfuscations $(i\mathcal{O}_{i,1}, \ldots i\mathcal{O}_{i,t})$ where $i\mathcal{O}_{i,j}$ is the obfuscation of the program $\mathsf{Prog}_{i,j}^{1,x_i,\rho_{x_i},r_{i,i},\rho_{r_{i,i}},\{Z_i\},\mathsf{fixedOutput}}$, where $\mathsf{fixedOutput}$ is the value $(m_{i,j}, \phi_{i,j})$ such that $\phi_{i,j}$ is the simulated proof that $m_{i,j}$ was generated correctly. (Recall that the flag has been set to 1 and this program on accepting inputs always outputs the value $\mathsf{fixedOutput}$.)
3. It sends $(\{r_{i,j}, \gamma_{i,j}\}_{j \in [n], j \neq i}, \{i\mathcal{O}_{i,j}\}_{j \in [t]})$ to $\mathcal{A}$ on behalf of P_i.

Round 2 Messages $\mathcal{A} \to \mathcal{S}$: Also in the second round the adversary $\mathcal{A}$ generates the messages on behalf of corrupted parties $\mathcal{P}^{\mathcal{A}}$. For each party $P_i \in \mathcal{P} \backslash \mathcal{P}^{\mathcal{A}}$ that has obtained "correctly formed" second round messages from all parties in $\mathcal{P}^{\mathcal{A}}$, our simulator sends $(\mathsf{generateOutput}, \mathsf{sid}, \mathcal{P}, P_i)$ to the ideal functionality.

This completes the description of the simulator.

Next we will prove via a sequence of hybrids that no environment $\mathcal{Z}$ can distinguish the ideal world experiment with $\mathcal{S}$ and $\mathcal{F}_f$ (as defined above) from a real execution of Π with $\mathcal{A}$. We will start with the real world execution in which the adversary $\mathcal{A}$ interacts directly with the honest parties holding their inputs and step-by-step make changes till we finally reach the simulator as described above. At each step will argue that the environment cannot distinguish the change except with negligible probability.

- H_1: This hybrid corresponds to the $\mathcal{Z}$ interacting with the real world adversary $\mathcal{A}$ and honest parties that hold their private inputs.

We can restate the above experiment with the simulator as follows. We replace the real world adversary $\mathcal{A}$ with the ideal world adversary $\mathcal{S}$. The ideal adversary $\mathcal{S}$ starts by invoking a copy of $\mathcal{A}$ and running a simulated interaction of $\mathcal{A}$ with the environment $\mathcal{Z}$ and the honest parties. $\mathcal{S}$ forwards the messages that $\mathcal{A}$ generates for it environment directly to $\mathcal{Z}$ and vice versa (as explained in the description of the simulator $\mathcal{S}$). In this hybrid the simulator $\mathcal{S}$ holds the private inputs of the honest parties and generates messages on their behalf using the honest party strategies as specified by Π.

• H_2: In this hybrid we change how the simulator generates the CRS. In particular we will change how $\mathcal{S}$ generates the public key pk of the CCA secure encryption scheme. We will not change the way CRS for the NIZK is generated.

$\mathcal{S}$ runs the setup algorithm $\mathsf{Gen}(1^\lambda)$ of the CCA secure encryption scheme and obtains a public key pk and a secret key sk. $\mathcal{S}$ will use this public key pk as part of the CRS and use the secret key sk to decrypt the ciphertexts generated by $\mathcal{A}$ on behalf of $\mathcal{P}^{\mathcal{A}}$. In particular for each party $P_i \in \mathcal{P}^{\mathcal{A}}$ our simulator proceeds as:

– Let $Z_i = \{c_i, \{d_{i,j}\}_{j\in[n]}\}$ be the message that $\mathcal{A}$ sends on behalf of P_i. Our simulator $\mathcal{S}$ decrypts the ciphertexts using the secret key sk. In particular $\mathcal{S}$ sets $x'_i = \mathsf{Dec}(sk, c_i)$ and $r'_{i,j} = \mathsf{Dec}(sk, d_{i,j})$. Obtain $x_i \in \{0,1\}^{\ell_{in}}$ such that $x'_i = i||x_i$. If x'_i is not of this form the set $x_i = \perp$. Similarly obtain $r_{i,j}$ from $r'_{i,j}$ for every j setting the value to $\perp$ in case it is not of the right format.

Note that in hybrid H_2 the simulator $\mathcal{S}$ additionally uses the secret key sk to extract the inputs of the adversarial parties. Furthermore if at any point in the execution any of the messages of the adversary are inconsistent with the input and randomness extracted but the adversary succeeds in providing an accepting NIZK proof then the simulator aborts, which event we call `Extract Abort`.

The distribution of the CRS, and hence the view of the environment $\mathcal{Z}$, in the two cases is identical. Also note that it follows from the perfect correctness of the encryption scheme and the statistical soundness of the NIZK proof system that the NIZK proofs adversary generates will have to be consistent with the extracted values. In other words over the random choices of the CRS we have that the probability of `Extract Abort` is negligible.

• H_3: In this hybrid we will change how the simulator generates the obfuscations on behalf of honest parties. Roughly speaking we observe that the obfuscations can only be evaluated to output one unique value (consistent with inputs and randomness extracted using sk) and we can just hardcode this value into the obfuscated circuit. More formally in the second round $\mathcal{S}$ generates the messages on behalf of the honest parties, i.e. parties in the set $\mathcal{P}\backslash\mathcal{P}^{\mathcal{A}}$ as follows:

1. For every P_j, $\mathcal{S}$ obtains $s_j = \oplus_{i=1}^{n} r_{i,j}$.
2. $\mathcal{S}$ virtually executes the protocol π with inputs $x_1, \ldots, x_n$ and random coins $s_1, \ldots, s_n$ for the parties $P_1, \ldots P_n$ respectively, and obtains the messages $m_{i,j}$ for all $i \in [n]$ and $j \in [t]$.
3. For each $P_i \in \mathcal{P}\backslash\mathcal{P}^{\mathcal{A}}$, $\mathcal{S}$ proceeds as follows:

(a) For every $j \in [n]$, $j \neq i$ generate $\gamma_{i,j}$ as a NIZK proof under σ for the NP-statement:

$$\left\{\exists\ \rho_{r_{i,j}} \ \middle|\ d_{i,j} = \mathsf{Enc}(i||r_{i,j}; \rho_{r_{i,j}})\right\}.$$

(b) A sequence of obfuscations $(i\mathcal{O}_{i,1}, \ldots i\mathcal{O}_{i,t})$ where $i\mathcal{O}_{i,j}$ is the obfuscation of the program $\mathsf{Prog}_{i,j}^{1,x_i,\rho_{x_i},r_{i,i},\rho_{r_{i,i}},\{Z_i\},\mathsf{fixedOutput}}$, where $\mathsf{fixedOutput}$ is the value $(m_{i,j}, \phi_{i,j})$ such that $\phi_{i,j}$ is the proof that $m_{i,j}$ was generated correctly. (Recall that the flag has been set to 1 and this program on all accepting inputs always outputs the value $\mathsf{fixedOutput}$.)

(c) It sends $(\{r_{i,j}, \gamma_{i,j}\}_{j\in[n],j\neq i}, \{i\mathcal{O}_{i,j}\}_{j\in[t]})$ to $\mathcal{A}$ on behalf of P_i.

We will now argue that hybrids H_2 and H_3 and computationally indistinguishable. More formally we will consider a sequence of $t \cdot |\mathcal{P}\backslash\mathcal{P}^{\mathcal{A}}|$ hybrids $H_{3,0,0}, \ldots H_{3,|\mathcal{P}\backslash\mathcal{P}^{\mathcal{A}}|,t}$. In hybrid $H_{3,i,j}$ all the obfusctaions by the first $i-1$ honest parties and the first j obfuscations generated by the i^{th} honest party are generated in the modified way as described above. It is easy to see that hybrid $H_{3,0,0}$ is same as hybrid H_2 and hybrid $H_{3,|\mathcal{P}\backslash\mathcal{P}^{\mathcal{A}}|,t}$ is same as hybrid H_3 itself.

We will now argue that the hybrids $H_{3,i,j-1}$ and $H_{3,i,j}$ for $j \in [t]$ are computationally indistinguishable. This implies the above claim, but in order to argue the above claim we first prove the following lemma.

Lemma 1.

$$\Pr\left[\begin{array}{l} \exists\ a, b\ : \\ \mathit{Prog}_{i,j}^{0,x_i,\rho_{x_i},r_{i,i},\rho_{r_{i,i}},\{Z_i\},0^{\ell_{i,j}}}(a) \neq \mathit{Prog}_{i,j}^{0,x_i,\rho_{x_i},r_{i,i},\rho_{r_{i,i}},\{Z_i\},0^{\ell_{i,j}}}(b) \\ \wedge\ \mathit{Prog}_{i,j}^{0,x_i,\rho_{x_i},r_{i,i},\rho_{r_{i,i}},\{Z_i\},0^{\ell_{i,j}}}(a) \neq \perp \\ \wedge\ \mathit{Prog}_{i,j}^{0,x_i,\rho_{x_i},r_{i,i},\rho_{r_{i,i}},\{Z_i\},0^{\ell_{i,j}}}(b) \neq \perp \end{array}\right] = \mathit{negl}(\lambda)$$

where the probability is taken over the random choices of the generation of the CRS.

Proof. Recall that program $\mathsf{Prog}_{i,j}^{0,x_i,\rho_{x_i},r_{i,i},\rho_{r_{i,i}},\{Z_i\},0^{\ell_{i,j}}}$ represents the j^{th} message function of the i^{th} party in protocol π. Recall that the input to the program consists of two $(R, \Gamma, M_{j-1}, \Phi_{j-1})$. We will refer to the (R, M_{j-1}) as the *main input part* and the Γ, Φ_{j-1} as the *proof part.*

Observe that since the proofs are always consistent with the extracted inputs and randomness, we have that there is a unique main input part for which adversary can provide valid (or accepting) proof parts. Further note that if the proof part is not accepting then $\mathsf{Prog}_{i,j}$ just outputs $\perp$. In other words if the proof is accepting then the program outputs a fixed value that depends just on the values that are fixed based on $\{Z_i\}$ values. We stress that the output actually does include a NIZK proof as well, however it is not difficult to see that this NIZK proof is also unique as a fixed randomness is used in generation of the proof. ∎

Armed with Lemma 1, we can conclude that the programs $\mathsf{Prog}_{i,j}^{0,x_i,\rho_{x_i},r_{i,i},\rho_{r_{i,i}},\{Z_i\},0^{\ell_{i,j}}}$ and $\mathsf{Prog}_{i,j}^{1,x_i,\rho_{x_i},r_{i,i},\rho_{r_{i,i}},\{Z_i\},\mathsf{fixedOutput}}$ are functionally equivalent. Next based on the indistinguisbaility obfuscation property, it is easy to see that the hybrids $H_{3,i,j-1}$ and $H_{3,i,j}$ are computationally indistinguishable.

• H_4: In this hybrid we change how the simulator generates the NIZKs on behalf of honest parties. Formally $\mathcal{S}$ generates the σ using the simulator S_1 of the NIZK proof system and generates all the proofs using the simulator S_2. The argument can be made formal by considering a sequence of hybrids and changing each of the NIZK proofs one at a time.

The indistinguishability between hybrids H_3 and H_4 can be based on the zero-knowledge property of the NIZK proof system.

• H_5: In this hybrid we change how the simulator $\mathcal{S}$ generates the first round messages on behalf of honest parties. In particular $\mathcal{S}$ instead of encrypting inputs and randomness of honest parties just encrypts zero strings of appropriate length.

We could try to base the indistinguishabilty between hybrids H_4 and H_5 on the semantic security of the PKE scheme. However observe that $\mathcal{S}$ at the same time should continue to be able to decrypt the ciphertexts that $\mathcal{A}$ generates on behalf of corrupted parties. Therefore we need to rely on the CCA security of the PKE scheme.

• H_6: In this hybrid instead of generating all the messages $m_{i,j}$ on behalf of honest parties honestly $\mathcal{S}$ uses $\mathcal{S}_\pi$ (the simulator for the underlying MPC protocol) to generated simulated messages.

The indistinguishability between hybrids H_5 and H_6 directly follows for the indistinguishability of honestly generated transcript in the execution of π from the transcript generated by $\mathcal{S}_\pi$.

• H_7: Observe that in hybrid H_6, $\mathcal{S}$ uses inputs of honest parties just in obtaining the output of the computation. It can obtain the same value by sending extracted inputs of the malicious parties to the ideal functionality $\mathcal{F}_f$.

Note that the hybrids H_6 and H_7 are identical. Observe that hybrid H_7 is identical to our simulator, which concludes the proof.