

# On Secure Two-Party Computation in Three Rounds

Prabhanjan Ananth<sup>1\*</sup> and Abhishek Jain<sup>2\*\*</sup>

<sup>1</sup> University of California Los Angeles

prabhanjan@cs.ucla.edu

<sup>2</sup> Johns Hopkins University

abhishek@cs.jhu.edu

**Abstract.** We revisit the exact round complexity of secure two-party computation. While four rounds are known to be sufficient for securely computing general functions that provide output to one party [Katz-Ostrovsky, CRYPTO'04], Goldreich-Krawczyk [SIAM J. Computing'96] proved that three rounds are insufficient for this task w.r.t. black-box simulation.

In this work, we study the feasibility of secure computation in three rounds using *non-black-box* simulation. Our main result is a three-round two-party computation protocol for general functions against adversaries with auxiliary inputs of a priori *bounded* size. This result relies on a new two round input-extraction protocol based on succinct randomized encodings.

We also provide a partial answer to the question of achieving security against non-uniform adversaries. Assuming sub-exponentially secure iO and one-way functions, we rule out three-round protocols that achieve polynomial simulation-based security against the output party and exponential indistinguishability-based security against the other party.

## 1 Introduction

The notion of secure computation [39,24] is fundamental in cryptography. Informally speaking, secure two-party computation allows two mutually distrusting parties to jointly compute a function over their private inputs in a manner such that no one learns anything beyond the function output.

An important measure of efficiency of secure computation protocols is *round complexity*. Clearly, the smaller the number of rounds, the lesser the impact of network latency on the communication between the parties. Indeed, ever since the introduction of secure computation, its round complexity has been the subject of intensive study, both in the two-party and multiparty setting.

In this work, we study the *exact* round complexity of secure two-party computation against malicious adversaries in the plain model (i.e., without any

---

\* Supported in part by grant 360584 from the Simons Foundation

\*\* Supported in part by a DARPA/ARL Safeware Grant W911NF-15-C-0213 and a sub-award from NSF CNS-1414023

trusted setup assumptions). We focus on the classical unidirectional message model where a round of communication consists of a single message sent by one party to the other.

In this setting, constant round protocols can be readily obtained by compiling a two-round semi-honest protocol (e.g., using garbled circuits [39] and oblivious transfer [37,15]) with constant-round zero-knowledge proofs [26,16,21] following the GMW paradigm [24]. Katz and Ostrovsky [30] established an upper bound on the exact round complexity of secure two-party computation by showing that four rounds are sufficient for computing general functions that provide output to one party. On the negative side, Goldreich and Krawczyk [22] proved that two-party computation with black-box simulation cannot be realized in three rounds.

Ever since the introduction of non-black-box techniques in cryptography nearly two decades ago [3], the following important question has remained open:

*Can secure two-party computation be realized in three rounds using non-black-box simulation?*

In this work, we address this question and provide both positive and negative results.

## 1.1 Our Results

We investigate the feasibility of three-round secure two-party computation against malicious adversaries in the plain model. We consider functions where only one party (a.k.a. *receiver*) learns the output. The other party is referred to as the *sender*.

**I. Positive Result.** Our main result is a three-round two-party computation protocol for general functions that achieves security against adversarial senders with auxiliary inputs of arbitrary polynomial size and adversarial receivers with auxiliary inputs of a priori *bounded* size.

In order to obtain our result, we devise a new non-black-box technique for extracting adversary’s input in only two rounds based on succinct randomized encodings [9,12,32] and two-round oblivious transfer (OT) with indistinguishability-based security [36]. To prove security of our three-round protocol, we additionally require two-message witness indistinguishable proofs (a.k.a. Zaps) [14] and Learning with Errors (LWE) assumption.

**Theorem 1.** *Assuming the existence of succinct randomized encodings, two-round OT, Zaps and LWE, there exists a three-round two-party computation protocol  $(P_1, P_2)$  for computing general functions that achieves security against adversarial  $P_1$  with auxiliary inputs of arbitrary polynomial size and adversarial  $P_2$  with auxiliary inputs of bounded size.*

*On Succinct Randomized Encodings.* A succinct randomized encoding (SRE) scheme allows one to encode the computation of a Turing machine  $M$  on an

input  $x$  such that the encoding time is independent of the time it takes to compute  $M(x)$ . The security of SRE is defined in a similar manner as standard (non-succinct) randomized encodings [28]. Presently, all known constructions of SRE are based on indistinguishability obfuscation (iO) [4,17]. We note, however, that SRE is *not* known to imply iO and may likely be a weaker assumption.<sup>3</sup>

*On Bounded Auxiliary Inputs.* Our positive result is motivated by the recent beautiful works of [8] and [7] on three-round zero-knowledge proofs that achieve security against adversaries with auxiliary inputs of a priori bounded size. Specifically, [8] considers malicious verifiers with bounded-size auxiliary inputs while [7] consider malicious provers with bounded-size auxiliary inputs.

Our positive result can be viewed as a generalization of [8] to general-purpose secure computation.

*Outputs for Both Parties.* Theorem 1 only considers functions that provide output to one party. As observed in [30], a protocol for this setting can be easily transformed into one where both parties receive the output by computing a modified functionality that outputs signed values. Now the output recipient can forward the output to the other party who accepts it only if the signature verifies.

**II. Negative Result.** We also explore the possibility of achieving security in the case where each adversarial party may receive auxiliary inputs of arbitrary polynomial size.

We provide a partial answer to this question. We show that three-round secure two-party computation for general functions is impossible if we require simulation-based security against PPT adversarial receivers and exponential indistinguishability security against adversarial senders. Our result relies on the existence of sub-exponentially secure iO and one-way functions.

**Theorem 2.** *Suppose that sub-exponentially secure iO and one-way functions exist. Then there exists a two-party functionality  $f$  such that no three-round protocol  $\Pi$  for computing  $f$  can achieve the following two properties:*

- *Simulation-based security against PPT adversarial receivers.*
- *$2^{O(L)}$ -indistinguishability security against adversarial senders, where  $L$  denotes the length of the first message in  $\Pi$ .*

Here,  $2^k$ -indistinguishability security means that for any pair of inputs  $(y, y')$  for the receiver, an adversarial sender can distinguish which input was used in a protocol execution with probability at most  $\frac{1}{2^k}$ .

We stress that Theorem 2 even rules out non-black-box simulation techniques.

*Discussion.* Our negative result can be viewed as a first step towards disproving the existence of three-round two-party computation against non-uniform adversaries. We remark that ruling out non-black-box techniques in three-rounds is

<sup>3</sup> If SRE satisfies an additional “output compactness” property where the size of an encoding of  $(M, x)$  is also independent of the size of the machine’s output, i.e.,  $|M(x)|$ , then sub-exponentially secure SRE is known to imply iO [2]. We do not require such output compactness property for our result.

highly non-trivial even when we require exponential (indistinguishability) security for one party. Indeed, a somewhat analogous question regarding the existence of three-round zero-knowledge *proofs* was recently addressed by Kalai et al. in [29]. Specifically, [29] prove the impossibility of three-round (public-coin) zero-knowledge proofs with non-black-box simulators assuming sub-exponentially secure iO and one-way functions and exponentially secure input-hiding obfuscation for multi-bit point functions.<sup>4</sup>

A proof system achieves *statistical* security against adversarial provers. In a similar vein, Theorem 2 requires exponential indistinguishability-security against adversarial senders. As such, Theorem 2 can be viewed as providing a complementary result to [29].

Needless to say, it remains an intriguing open question to extend our lower bound to rule out protocols that achieve polynomial-security against adversarial senders.

## 1.2 Our Techniques

In this section, we describe the main ideas used in our positive and negative results.

**I. Positive Result.** We start by describing the main ideas in our positive result. We first describe the setting: we consider two parties  $P_1$  and  $P_2$  holding private inputs  $x_1$  and  $x_2$ , respectively, for computing a function  $f$ . At the end of the protocol,  $P_2$  gets  $f(x_1, x_2)$  while  $P_1$  gets no output. We want to achieve security against adversarial  $P_1$  who may receive auxiliary inputs of unbounded (polynomial) size and adversarial  $P_2$  who may receive auxiliary inputs of an a priori bounded size.

Recently, Bitansky et. al. [8] constructed a three-round zero-knowledge argument of knowledge (ZKAOK) that achieves standard soundness guarantee and zero-knowledge guarantee against adversarial verifiers with bounded auxiliary inputs. Given their protocol, a natural starting idea to achieve our goal is to “compile” a two-round semi-honest two-party computation protocol into a maliciously secure one (a la [24]) with their ZKAOK system. Note, however, that while we have enough rounds in the protocol to enforce semi-honest behavior on  $P_1$  using ZKAOK, we cannot use the same approach for  $P_2$ . Nevertheless, as a first step, let us fix a three-round protocol that guarantees security against adversarial  $P_1$ . For concreteness, we instantiate the semi-honest two-party computation using garbled circuits and two-round oblivious transfer. We also use a delayed-input ZKAOK [33] where the instance is only used in the last round. This property is satisfied by argument system of [8].

- In the first round,  $P_1$  sends the first message of a delayed-input ZKAOK.
- In the second round,  $P_2$  sends the second message of ZKAOK together with the receiver message of a two-round oblivious transfer (OT) computed using its input for  $f$ .

<sup>4</sup> Their result, in fact, extends to constant-round protocols.

- In the third round,  $P_1$  sends garbled circuit for  $f$  with its input hardwired, together with the OT sender message (computed using the inputs labels for the garbled circuit) and the third message of ZKAOK to prove that the garbled circuit and the OT sender message are computed “honestly”.

*Main Challenge #1.* Note that in the above protocol, it is already guaranteed that  $P_2$ ’s input is independent of  $P_1$ ’s input. Nevertheless, this is not enough and in order to achieve security against malicious  $P_2$ , we need to construct a polynomial-time simulator that can *extract*  $P_2$ ’s input by the end of the second round, and then simulate the third round of the protocol to “force” the correct output on  $P_2$ . In light of our lower bound, we need to devise a two-round input extraction procedure that works against adversaries with bounded auxiliary inputs. At first, it is not at all clear how such an input-extraction protocol can be constructed. In particular, black-box techniques do not suffice for this purpose [22]. Instead, we must use non-black-box techniques.

The problem of extraction in two-rounds or less was recently considered by Bitansky et al. [8]. They study extractable one-way functions and then use them to construct three-round ZKAOK against verifiers with bounded non-uniformity. We note, however, that their notion of extractable one-way functions is unsuitable for our goal of extracting adversary’s input. In particular, in their notion, the extracted value can be from a completely different distribution than the actual value  $x$  used to compute the one-way function. In contrast, we want to extract a “committed” input of the adversary.

*Main Challenge #2.* To make matters worse, we cannot hope to extract the input of a malicious adversary in two rounds with guarantee of correct extraction. Indeed, two-round zero-knowledge proofs (with polynomial-time simulation) are known to be impossible against non-uniform verifiers even w.r.t. non-black-box simulation [25].<sup>5</sup>

In light of the above, we settle on a “weak extraction” guarantee, namely, where correctness of extraction is only guaranteed if the adversary behaves honestly. Note that this means that our simulator may fail to extract the input of  $P_2$  if it behaves maliciously. In this case, it may not be able to produce an indistinguishable third message of the protocol.

For now, we ignore this important issue and proceed to describe a two-round protocol that enables weak input-extraction. Later, we describe how we construct our scheme using only this weak extraction property.

*(Weak) Input-Extraction in Two Rounds.* We want to construct a two-round protocol that allows a simulator (that has access to the Turing machine description and bounded auxiliary input of adversarial  $P_2$ ) to extract  $P_2$ ’s input for  $f$  as

---

<sup>5</sup> Bitansky et al. [8] construct a two-round zero-knowledge argument against verifiers with bounded non-uniformity. Using their system, however, would necessarily require even  $P_1$  (who will play the role of the verifier) to have bounded non-uniformity. Our goal instead is to limit the bounded non-uniformity assumption to  $P_2$  and allow  $P_1$  to be fully non-uniform.

long as  $P_2$  behaves semi-honestly in this protocol. However, an adversarial  $P_1$  should not be able to learn any information about an honest  $P_2$ 's input. For simplicity of exposition, below, we restrict ourselves to the case where  $P_2$  is a uniform Turing machine. It is easy to verify that our protocol also works when  $P_2$  has an auxiliary input of bounded length.

We first note that the problem of constructing an input-extraction protocol can be reduced to the problem of constructing a “trapdoor” extraction protocol where the trapdoor is a random string. This is because the trapdoor can be set to the randomness  $r$  used by  $P_2$  for computing its OT receiver message in our three-round protocol described earlier. If we use an OT protocol where the receiver’s message is perfectly binding (e.g., [36]), then once the simulator has extracted  $P_2$ 's randomness in OT, it can also recover its input.

In order to construct a trapdoor extraction protocol, we build on ideas from Barak’s non-black-box technique [3]. Consider the following two-party functionality  $g$ : it takes as input a string TM from  $P_1$  and a tuple  $(\beta, \text{trap}, m)$  from  $P_2$ . It treats TM as a valid Turing machine and computes  $\beta' = \text{TM}(m)$ . If  $\beta' = \beta$ , it outputs  $\text{trap}$ , else it outputs  $\perp$ .<sup>6</sup> Let  $\Pi$  be a two-round two-party computation protocol for computing  $g$ .

Now, consider the following candidate two-round protocol for extracting a trapdoor from  $P_2$ :  $P_1$  sends the first message of  $\Pi$  computed using input  $\text{TM} = 0$ . Let  $\text{msg}_1$  denote this message. Upon receiving  $\text{msg}_1$ ,  $P_2$  first prepares an input tuple  $(\beta, \text{trap}, m)$  for  $g$  as follows: it samples a random string  $\beta$  of length  $\ell$  s.t.  $\ell \gg |\text{msg}_1|$  and sets  $\text{trap}$  to be a random string and  $m = \text{msg}_1$ . Finally,  $P_2$  sends the second message of  $\Pi$  computed using  $(\beta, \text{trap}, m)$  together with  $\beta$ .

A non-black-box simulator that knows the Turing machine description  $\text{TM}_2$  of adversarial  $P_2$  can set its input  $\text{TM} = \text{TM}_2$  in the above protocol. If  $P_2$  behaves semi-honestly, then at the end of the protocol, the simulator should obtain  $\text{trap}$ . Security against a malicious  $P_1$  can be argued using the fact that  $\beta \gg |\text{msg}_1|$  in the same manner as the proof of soundness in Barak’s protocol.

A reader familiar with [3] may notice a major problem with the above extraction protocol. Note that since  $\Pi$  is a secure computation protocol, its running time must be strictly greater than the size of the circuit representation of  $g$ . Now, since the functionality  $g$  internally computes the next-step function of  $P_2$ , the running time of  $\Pi$  is strictly greater than the running time of  $P_2$ !

Our key idea to solve this problem is to *delegate* the “expensive” computation inside  $g$  to  $P_1$  (or more accurately, the simulator when  $P_2$  is corrupted).<sup>7</sup> Let  $M$  be an “input-less” Turing machine that has hardwired in its description a tuple  $(\text{TM}, \beta, \text{trap}, m)$ . Upon execution, it performs the same computation as  $g$ . Now, instead of using the two-party computation protocol to compute the function  $g$ , we use it to compute a “secure encoding” of  $M$ . We want the encoding scheme

<sup>6</sup> Note that  $g$  internally transforms TM into a circuit and uses it to perform the rest of the computation.

<sup>7</sup> Indeed, an honest  $P_1$  is never required the functionality  $g$ . However, when  $P_2$  is corrupted, then the simulator acting on behalf of  $P_1$  does compute  $g$  to learn the trapdoor.

to be such that the time to encode  $M$  is *independent* of the running time of  $M$ . Note that in this case, the running time of the protocol is also independent of the running time of  $M$ . The honest  $P_1$  ignores the encoding it obtains at the end of the two-party computation protocol. However, the simulator can simply “decode” the secure encoding to learn its output.

An encoding scheme with the above efficiency property is referred to as a succinct randomized encoding (SRE) [9,12,32]. By using an SRE scheme, we are able to resolve the running-time problem.

*Using Weak Extraction Guarantee.* Finally, we explain how we obtain our construction by only relying on the weak extraction property of our input extraction protocol. Note that if an adversarial  $P_2$  cheats in the input extraction protocol, then due to the weak extraction guarantee, the simulator may extract an incorrect input (or no input at all). In this case, the simulated garbled circuit computed by the simulator would be easily distinguishable from the garbled circuit in the real execution. Therefore, we need a mechanism that “hides”  $P_1$ ’s third round message from  $P_2$  if  $P_2$  cheated in the input-extraction protocol. On the other hand, if  $P_2$  did behave honestly, then the mechanism should “reveal” the third round message to  $P_2$ .

We solve this problem by using conditional disclosure of secrets [19,1]. Recall that a CDS scheme consists of two players: a sender  $S$  and a receiver  $R$ . The parties share a common instance  $x$  of an NP language. Using this instance, the sender  $S$  can “encrypt” a secret message  $m$  s.t. a receiver  $R$  can only “decrypt” it using a witness  $w$  for  $x$ .

Using a CDS scheme for NP, we modify our protocol as follows. Now,  $P_1$  will send a CDS encryption of the garbled circuit for  $f$  and its OT sender message. The instance for this encryption is simply the transcript of the input extraction protocol. In order to decrypt,  $P_2$  must use a witness that establishes honest behavior during the input extraction protocol. The input and randomness of  $P_2$  in the input-extraction protocol constitutes such a witness. In other words, if  $P_2$  cheated in the input-extraction protocol, then it cannot recover the third round message of  $P_1$ .

A subtle point here is that a CDS scheme only promises security against adversarial receivers when the instance used for encryption is *false*. Therefore, in order to use the security of CDS, we must ensure that there does *not exist* a valid witness if  $P_2$  cheats in the input extraction protocol. We achieve this property by ensuring that the input-extraction protocol is perfectly binding for  $P_2$ .

We implement a CDS scheme using a two-round two-party computation protocol that achieves indistinguishability security against malicious receivers and semi-honest senders. Such a scheme can be implemented using garbled circuits and two-round oblivious transfer of [36]. Finally, to prevent an adversarial  $P_1$  from creating “malformed” CDS encryptions, we require  $P_1$  to prove its well-formedness using delayed-input ZKAOK.



**II. Negative Result.** We now provide an overview of our lower bound. Due to space constraints, we describe the lower bound in the full version.

Recall that simulation-based security for any two-party computation protocol is argued by constructing a polynomial-time simulator who can simulate the view of the adversary in an indistinguishable manner without any knowledge of the honest party input. One of the main tasks of such a simulator is to *extract* the input of the adversary. We establish our negative result by ruling out the possibility of extracting the input of adversarial receiver in a three-round secure computation protocol.

More concretely, we consider three round protocols  $(P_1, P_2)$  where  $P_2$  receives the output. We describe a two-party functionality  $f$  and an adversary  $P_2$  such that no polynomial-time simulator can extract  $P_2$ 's input from any three-round protocol  $\Pi$  for computing  $f$ , if  $\Pi$  achieves  $2^{O(L)}$ -indistinguishability security against  $P_1$ . Here,  $L$  is the length of the first message of  $\Pi$ .

Note that in a three-round protocol,  $P_2$  only sends a single message. Clearly, black-box techniques are insufficient for extracting  $P_2$ 's input in this setting. The main challenge here is to rule out extraction via *non-black-box* techniques.

In order to “hide” the input of an adversarial  $P_2$  from a non-black-box simulator who has access to  $P_2$ 's code, we make use of program obfuscation [4]. Namely, we construct a “dummy” adversary  $P_2$ , who receives as *auxiliary input*, an obfuscated program that has an input hardwired in its description and uses it to compute the adversary’s message in the two-party computation protocol. During the protocol execution, the adversary simply uses the obfuscated program to compute its protocol message. Our goal is to then argue that having access to the code of this dummy adversary as well as his obfuscated auxiliary input gives no advantage to a polynomial-time simulator. We note that a similar strategy was recently used by Bitansky et al. [8] in order to prove the impossibility of extractable one-way functions.

Below, we first describe our proof strategy using the strong notion of virtual black-box obfuscation [4]. Most of the main challenges that we address already arise in this case. Later, we explain how we can derive our negative result using the weaker notion of indistinguishability obfuscation.

*Function  $f$ .* Recall that the main reason why the simulator needs to extract the adversary’s input is to learn the function output from the ideal functionality. In order to ensure that the simulator cannot “bypass” input extraction, we choose a function with unpredictable outputs. Furthermore, we also want that the input of the honest party cannot be trivially determined from the function.

We choose  $f$  to be a pseudorandom function PRF that takes as input a PRF key  $x_1$  from  $P_1$  and an input  $x_2$  from  $P_2$  and outputs the evaluation of the PRF on  $x_2$  using key  $x_1$ . It is easy to see that  $f$  satisfies the above desired properties.

*Adversary  $P_2$  and Auxiliary Input  $Z$ .* Towards a contradiction, let  $\Pi$  be any three-round two-party protocol for securely computing  $f$  with the security properties stated in Theorem 2.

The auxiliary input  $Z$  consists of an obfuscated program that has an input  $x_2$  and a key  $K$  hardwired in its description:



1. Upon receiving a message  $\text{msg}_1$  from  $P_1$  as input, the program honestly computes the protocol message  $\text{msg}_2$  of  $P_2$  (as per protocol  $\Pi$ ) using input  $x_2$  and randomness  $r = F(K, \text{msg}_1)$ , where  $F$  is another PRF.
2. Upon receiving a protocol transcript  $(\text{msg}_1, \text{msg}_2, \text{msg}_3)$ , it re-computes the randomness  $r$  used to compute  $\text{msg}_2$ . Using the transcript, randomness  $r$  and input  $x_2$ , it computes the output honestly.

The adversary  $P_2$  does not perform any computation on its own. Upon receiving a message  $\text{msg}_1$  from  $P_1$ , it runs the obfuscated program on  $\text{msg}_1$  to obtain  $\text{msg}_2$  and then forwards it to  $P_1$ . Finally, upon receiving  $\text{msg}_3$  from  $P_1$ , it submits the protocol transcript  $(\text{msg}_1, \text{msg}_2, \text{msg}_3)$  to the obfuscated program to obtain an output  $y$ .

*Proof Strategy: Attempt #1.* For any simulator  $S$  for  $\Pi$ , let  $Q$  denote the possible set of queries made by  $S$  to the ideal function. The core argument in our proof is that the query set  $Q$  *cannot* contain  $P_2$ 's input  $x_2$ . At a high-level, our strategy for proving this is as follows: first, we want to switch the auxiliary input  $Z$  to a different auxiliary input  $Z'$  that has some other input  $x'_2$  hardwired inside it. We want to rely upon the security of  $\Pi$  against adversarial  $P_1$  in order to make this switch. Once we have made this switch, then we can easily argue that the  $Q$  cannot contain  $x_2$  since the view of  $S$  is independent of  $x_2$ .

*Problem: Rewinding Attacks.* The above proof strategy runs into the following issue: since the adversary  $P_2$  includes the protocol output in its view, a simulator  $S$  may fix the first two messages of the protocol and then try to observe the output of  $P_2$  on many different third messages. Indeed, a simulator may be able to learn non-trivial information by simply observing whether the adversary accepts or aborts on different trials.

A naive approach to try to address this problem is to simply remove the output from adversary's view. That is, we simply delete the second instruction in the obfuscated program  $Z$ . Now,  $P_2$  never processes the messages received from  $P_1$ . This approach, however, immediately fails because now a simulator can simply simulate a "rejecting" transcript. Since there is no way for the distinguisher to check the validity of the transcript (since  $P_2$ 's output is not part of its view), the simulator can easily fool the distinguisher.

*Non-uniform Distinguishers.* We address this problem by using non-uniform distinguishers, in a manner similar to [25]. Specifically, we modify  $P_2$  to be such that it simply outputs the protocol transcript at the end of the protocol. The PRF key  $K$  hardwired inside  $Z$  (and used to compute  $P_2$ 's protocol message) is given as non-uniform advice to the distinguisher. Note that this information is not available to the simulator.

Now, given  $K$  and the protocol transcript, the distinguisher can easily compute  $P_2$ 's output. Therefore, a simulator can no longer fool the distinguisher via a rejecting transcript. Furthermore, now, the protocol output is not part of  $P_2$ 's view, and therefore, rewinding attacks are also ruled out.

*Revised Proof Strategy.* Let us now return to our proof strategy. Recall that we want to switch the auxiliary input  $Z$  to a different auxiliary input  $Z'$  that has

some other input  $x'_2$  hardwired inside it. Once we have made this switch, then we can easily argue that the  $Q$  cannot contain  $x_2$  since the view of  $S$  is independent of  $x_2$ .

We make the switch from auxiliary input  $Z$  to  $Z'$  via a sequence of hybrids. In particular, we go through  $2^L$  number of hybrids, one for every possible first message  $\text{msg}_1$  of  $P_1$ . In the  $i^{\text{th}}$  hybrid, we use an auxiliary input  $Z_i$  that has both  $x_2$  and  $x'_2$  hardwired inside it. On input first messages  $\text{msg}_1 < i$ , it uses  $x_2$  to compute the second message, and otherwise, it uses  $x'_2$ . In order to argue indistinguishability of hybrids  $i$  and  $i + 1$ , we use the security of protocol  $\Pi$  against malicious  $P_1$ . Indeed, this is why we require  $2^{O(L)}$ -indistinguishability security against adversarial  $P_1$ .

In order to perform the above proof strategy using indistinguishability obfuscation (as opposed to virtual black-box obfuscation), we make use of puncturable PRFs and use the ‘‘punctured programming’’ techniques [38] that have been used in a large body of works over the last few years. We refer the reader to the technical sections for further details.

### 1.3 Related Works

Katz and Ostrovsky [30] constructed a four-round two-party computation protocol for general functions where one of the parties receives the output. Recently, Garg et al. [18] extended their work to the simultaneous-message model.

Three round zero-knowledge proofs were first constructed in [27,6] using ‘‘knowledge assumptions.’’ More recently, [8,7] construct three-round zero-knowledge proofs adversaries that receive auxiliary inputs of a priori bounded size. Our positive result is directly inspired by these works.

A recent work of Döttling et al. [13] constructs a two-round two-party computation protocol for oblivious computation of cryptographic functionalities. They consider semi-honest senders and malicious receivers, and prove game-based security against the latter. In contrast, in this work, we consider polynomial-time simulation-based security.

## 2 Preliminaries

We denote the security parameter by  $\lambda$ . We assume familiarity with standard cryptographic primitives.

*General Notation.* If  $A$  is a probabilistic polynomial time algorithm, then we write  $y \leftarrow A(x)$  to denote that one execution of  $A$  on  $x$  yields  $y$ . Furthermore, we denote  $y \leftarrow A(x; r)$  to denote that  $A$  on input  $x$  and randomness  $r$ , outputs  $y$ . If  $\mathcal{D}$  is a distribution, we mean  $x \xleftarrow{\$} \mathcal{D}$  to mean that  $x$  is sampled from  $\mathcal{D}$ .

Two distributions  $\mathcal{D}_1$  and  $\mathcal{D}_2$ , defined on the same sample space, are said to be computationally distinguishable, denoted by  $\mathcal{D}_1 \not\cong_{c,\epsilon} \mathcal{D}_2$  if the following

holds: for any PPT adversary  $\mathcal{A}$  and sufficiently large security parameter  $\lambda \in \mathbb{N}$  it holds that,

$$|\Pr[1 \leftarrow \mathcal{A}(1^\lambda, s_1) : s_1 \xleftarrow{\$} \mathcal{D}_1(1^\lambda)] - \Pr[1 \leftarrow \mathcal{A}(1^\lambda, s_2) : s_2 \xleftarrow{\$} \mathcal{D}_2(1^\lambda)]| \leq \varepsilon,$$

If  $\varepsilon$  is some negligible function then we denote this by  $\mathcal{D}_1 \cong_c \mathcal{D}_2$ .

*Languages and Relations.* A language  $L$  is a subset of  $\{0, 1\}^*$ . A relation  $\mathcal{R}$  is a subset of  $\{0, 1\}^* \times \{0, 1\}^*$ . We use the following notation:

- Suppose  $\mathcal{R}$  is a relation. We define  $\mathcal{R}$  to be *efficiently decidable* if there exists an algorithm  $A$  and fixed polynomial  $p$  such that  $(x, w) \in \mathcal{R}$  if and only if  $A(x, w) = 1$  and the running time of  $A$  is upper bounded by  $p(|x|, |w|)$ .
- Suppose  $\mathcal{R}$  is an efficiently decidable relation. We say that  $\mathcal{R}$  is a NP relation if  $L(\mathcal{R})$  is a NP language, where  $L(\mathcal{R})$  is defined as follows:  $x \in L(\mathcal{R})$  if and only if there exists  $w$  such that  $(x, w) \in \mathcal{R}$  and  $|w| \leq p(|x|)$  for some fixed polynomial  $p$ .

*Modeling Real World Adversaries: Uniform versus Non Uniform.* One way to model real world adversaries  $\mathcal{A}$  is by representing them as a class of non uniform circuits  $\mathcal{C}$ , one circuit per input length. This is the standard definition of adversaries considered in the literature. We call such adversaries *non uniform* adversaries.

Yet another type of adversaries are  *$\mu$ -bounded uniform* adversaries: in this case, the real world  $\mathcal{A}$  is represented by a probabilistic Turing machine  $M$  and can additionally receive as input auxiliary information of length at most  $\mu(\lambda)$ . The description size of  $\mathcal{A}$  is the sum total of the description size of  $M$  and  $\mu(\lambda)$ . We say that  $\mathcal{A}$  is uniform if it does not receive any additional auxiliary information. In this case, the description size of  $\mathcal{A}$  is nothing but the description size of the Turing machine representing  $\mathcal{A}$ .

*Notation for Protocols.* Consider a two party protocol  $\Pi$  between parties  $P_1$  and  $P_2$ . We define the notation  $P_1.\text{MsgGen}[\Pi]$  (resp.,  $P_2.\text{MsgGen}[\Pi]$ ) to denote the algorithm that generates the next message of  $P_1$  (resp.,  $P_2$ ). The notation  $\beta \leftarrow P_1.\text{MsgGen}[\Pi](\alpha, \text{st}; r)$  indicates that the output of next message algorithm of party  $P_1$  on input  $\alpha$ , current state  $\text{st}$  and randomness  $r$  is the string  $\beta$ . Initially,  $\text{st}$  is set to  $\perp$ . For convenience of notation, we assume that the  $\text{MsgGen}[\cdot]$  is a stateful algorithm and hence, we avoid describing the parameter  $\text{st}$  explicitly.

We denote the view of a party in a secure protocol to consist of its input, randomness and the transcript of messages exchanged by the party. For a party  $P$  with input  $y$  (that includes randomness), we denote its view by  $\text{View}_{P,y}$ .

## 2.1 Secure Two-Party Computation

A secure two-party computation protocol is carried out between two parties  $P_1$  and  $P_2$  (modeled as interactive Turing machines) and is associated with a deterministic functionality  $f$ . Party  $P_1$  has input  $x_1$  and  $P_2$  has input  $x_2$ . At the end of the protocol,  $P_2$  gets the output.

*Simulation-based Security.* We follow the real/ideal world paradigm to formalize the security of a two party computation protocol  $\Pi_{2PC}$  secure against malicious adversaries.<sup>8</sup> We follow the description presented in Lindell-Pinkas [34]. First, we begin with the ideal process.

IDEAL PROCESS: The ideal world is associated with a trusted party and parties  $P_1, P_2$ . At most one of  $P_1, P_2$  is controlled by an adversary<sup>9</sup>. The process proceeds in the following steps:

1. **Input Distribution:** The environment distributes the inputs  $x_1$  and  $x_2$  to parties  $P_1$  and  $P_2$  respectively.
2. **Inputs to Trusted Party:** The parties now send their inputs to the trusted party. The honest party sends the same input it received from the environment to the trusted party. The adversary, however, can send a different input to the trusted party.
3. **Aborting Adversaries:** An adversarial party can then send a message to the trusted party to abort the execution. Upon receiving this, the trusted party terminates the ideal world execution. Otherwise, the following steps are executed.
4. **Trusted party answers party  $P_2$ :** Suppose the trusted party receives inputs  $x'_1$  and  $x'_2$  from  $P_1$  and  $P_2$  respectively. It sends the output  $\mathbf{out} = f(x'_1, x'_2)$  to  $P_2$ .
5. **Output:** If the party  $P_2$  is honest, then it outputs  $\mathbf{out}$ . The adversarial party ( $P_1$  or  $P_2$ ) outputs its entire view.

We denote the adversary participating in the above protocol to be  $\mathcal{B}$  and the auxiliary input to  $\mathcal{B}$  is denoted by  $\mathbf{z}$ . We define  $\mathbf{Ideal}_{f, \mathcal{B}}^{\Pi_{2PC}}(x_1, x_2, \mathbf{z})$  to be the joint distribution over the outputs of the adversary and the honest party<sup>10</sup>.

REAL PROCESS: In the real process, both the parties execute the protocol  $\Pi_{2PC}$ . At most one of  $P_1, P_2$  is controlled by an adversary. We denote the adversarial party to be  $\mathcal{A}$ . As in the ideal process, they receive inputs from the environment. We define  $\mathbf{Real}_{f, \mathcal{P}}^{\Pi_{2PC}}(x_1, x_2, \mathbf{z})$  to be the joint distribution over the outputs of the adversary and the honest party, where  $\mathbf{z}$  denotes the auxiliary information.

We define the security of two party computation as follows:

**Definition 1 (Security).** *Consider a two party functionality  $f$  as defined above. Let  $\Pi_{2PC}$  be a two party protocol implementing  $f$ . We say that  $\Pi_{2PC}$  **securely***

<sup>8</sup> Malicious adversaries can arbitrarily deviate from the protocol. The other type of adversaries commonly considered are semi-honest adversaries, where the adversaries follow the protocol but try to gain information by observing the conversation with the honest party. Both type of adversaries are allowed to substitute the inputs they receive from the external environment with inputs of their choice.

<sup>9</sup> This means that at most one of the parties could deviate from the rules prescribed by the ideal process.

<sup>10</sup> If  $P_1$  is honest, it does not have any output.

**computes**  $f$  if for every PPT malicious adversary  $\mathcal{A}$  in the real world, there exists a PPT adversary  $\mathcal{B}$  in the ideal world such that: for every auxiliary information  $\mathbf{z} \in \{0, 1\}^{\text{poly}(\lambda)}$ ,

$$\text{Ideal}_{f, \mathcal{B}}^{\Pi_{2\text{PC}}}(x_1, x_2, \mathbf{z}) \cong_c \text{Real}_{f, \mathcal{A}}^{\Pi_{2\text{PC}}}(x_1, x_2, \mathbf{z})$$

In this work, we are interested in the setting when the adversary corrupting  $P_2$  (who receives the output) in the above protocol is  $\mu$ -uniform. We allow for adversarial  $P_1$  to be non-uniform. We formally define this below.

**Definition 2 (Security Against  $\mu$ -Bounded Uniform  $P_2$ ).** Consider a two party functionality  $f$  as defined above. Let  $\Pi_{2\text{PC}}$  be a two party protocol computing  $f$ . We say that  $\Pi_{2\text{PC}}$  **securely computes**  $f$  if the following holds:

- For every  $\mu$ -bounded uniform malicious adversary  $\mathcal{A}$  in the real world corrupting party  $P_2$ , there exists a PPT adversary  $\mathcal{B}$  in the ideal world such that: for every auxiliary information  $\mathbf{z} \in \{0, 1\}^{\mu(\lambda)}$ ,

$$\text{Ideal}_{f, \mathcal{B}}^{\Pi_{2\text{PC}}}(x_1, x_2, \mathbf{z}) \cong_c \text{Real}_{f, \mathcal{A}}^{\Pi_{2\text{PC}}}(x_1, x_2, \mathbf{z})$$

- For every PPT non-uniform malicious adversary  $\mathcal{A}$  in the real world corrupting  $P_1$ , there exists a PPT adversary  $\mathcal{B}$  in the ideal world such that: for every auxiliary information  $\mathbf{z} \in \{0, 1\}^{\text{poly}(\lambda)}$ ,

$$\text{Ideal}_{f, \mathcal{B}}^{\Pi_{2\text{PC}}}(x_1, x_2, \mathbf{z}) \cong_c \text{Real}_{f, \mathcal{A}}^{\Pi_{2\text{PC}}}(x_1, x_2, \mathbf{z})$$

### 3 Building Blocks

We describe the building blocks used in our results.

#### 3.1 Garbling Schemes

We recall the definition of garbling schemes [39, 5].

**Definition 3 (Garbling Schemes).** A garbling scheme  $\text{GC} = (\text{Gen}, \text{GrbC}, \text{Grbl}, \text{EvalGC})$  defined for a class of circuits  $\mathcal{C}$  consists of the following polynomial time algorithms:

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

It satisfies the following properties:

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

$$\Pr \left[ C(x) \leftarrow \text{EvalGC}(\widehat{C}, (k_1^{x_1}, \dots, k_\ell^{x_\ell})) : \begin{array}{l} \text{gcsk} \leftarrow \text{Gen}(1^\lambda), \\ \widehat{C} \leftarrow \text{GrbC}(\text{gcsk}, C), \\ ((k_1^0, k_1^1), \dots, (k_\ell^0, k_\ell^1)) \leftarrow \text{Grbl}(\text{gcsk}) \end{array} \right] = 1$$

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

$$\left\{ (\widehat{C}, k_1^{x_1}, \dots, k_\ell^{x_\ell}) \right\} \cong_c \left\{ \text{Sim}(1^\lambda, \phi(C), C(x)) \right\},$$

where:

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

**Theorem 3 ([39]).** *Assuming one-way functions, there exists a secure garbling scheme.*

*Deterministic Garbling.* For our results, we need a garbling scheme where the circuit garbling algorithms and the garbling key generation algorithms are deterministic. Any garbling scheme can be transformed into one satisfying these properties by generating a PRF key as part of the setup algorithm. The randomness in the circuit garbling and the garbling key generation algorithms can be derived from the PRF key.

### 3.2 Oblivious Transfer

We recall the notion of oblivious transfer [37,15] below. We adopt the indistinguishability security notion. Against malicious senders, indistinguishability security says that a malicious sender should not be able to distinguish the receiver’s input. Defining security against malicious receivers is more tricky, we require that if  $c$  is the choice bit committed to by the receiver then the receiver should get no information about the bit  $b_{\bar{c}}$  in the pair  $(b_0, b_1)$ , where  $(b_0, b_1)$  is the pair of bits used by the honest sender. This is formalized by using unbounded extraction.

**Definition 4 (Oblivious Transfer).** *A 1-out-2 oblivious transfer (OT) protocol OT is a two party protocol between a sender and a receiver. A sender has two input bits  $(b_0, b_1)$  and the receiver has a choice bit  $c$ . At the end of the protocol, the receiver receives an output bit  $b'$ . We denote this process by  $b' \leftarrow \langle \text{Sen}(b_0, b_1), \text{Rec}(c) \rangle$ .*

*We require that an OT protocol satisfies the following properties:*

– **Correctness:** For every  $b_0, b_1, c \in \{0, 1\}$ , we have:

$$\Pr[b_c \leftarrow \langle \text{Sen}(b_0, b_1), \text{Rec}(c) \rangle] = 1$$

– **Indistinguishability security against malicious senders:** For all PPT senders  $\text{Sen}^*$ , for all auxiliary information  $\mathbf{z} \in \{0, 1\}^*$  we have,

$$|\Pr[1 \leftarrow \langle \text{Sen}^*(\mathbf{z}), \text{Rec}(0) \rangle] - \Pr[1 \leftarrow \langle \text{Sen}^*(\mathbf{z}), \text{Rec}(1) \rangle]| \leq \frac{1}{2} + \text{negl}(\lambda).$$

– **Indistinguishability Security against malicious receivers:** For all PPT receivers  $\text{Rec}^*$ , we require that the following holds. There exists an extractor  $\text{Ext}$  (not necessarily efficient) that extracts a bit from the view of  $\text{Rec}^*$  such that the following holds: For any auxiliary information  $\mathbf{z} \in \{0, 1\}^*$ ,

$$\begin{aligned} & |\Pr[1 \leftarrow \langle \text{Sen}(\{b_c, b_{\bar{c}}\}_{c \in \{0, 1\}}), \text{Rec}^*(\mathbf{z}) \rangle \mid c \leftarrow \text{Ext}(\text{View}_{\text{Rec}^*, \mathbf{z}})] \\ & - \Pr[1 \leftarrow \langle \text{Sen}(\{b_c, \bar{b}_{\bar{c}}\}_{c \in \{0, 1\}}), \text{Rec}^*(\mathbf{z}) \rangle \mid c \leftarrow \text{Ext}(\text{View}_{\text{Rec}^*, \mathbf{z}})]| \leq \frac{1}{2} + \text{negl}(\lambda). \end{aligned}$$

We define  $\ell$ -parallel 1-out-2 OT to be a protocol that is composed of  $\ell$  parallel executions of 1-out-2 OT protocol.

For our main result, we require an oblivious transfer protocol that satisfies the following additional property.

**Definition 5 (Uniqueness of Transcript).** Consider an 1-out-2 oblivious transfer protocol  $\text{OT}$  between two parties  $P_1$  (sender) and  $P_2$  (receiver). We say that  $\text{OT}$  satisfies **uniqueness of transcript property** if the following holds: Consider an execution of  $P_1(b_0, b_1; r_1)$  and  $P_2(c; r_2)$  and let the transcript of the execution be denoted by  $\text{Transcript} = (OT_1, \dots, OT_k)$ . Suppose there exists  $c' \in \{0, 1\}$  and string  $r'_2$  such that the execution of  $P_1(b_0, b_1; r_1)$  and  $P_2(c'; r'_2)$  leads to the same transcript  $\text{Transcript}$  then it should hold that  $c' = c$  and  $r_2 = r'_2$ . Also it follows that, given  $r_2$ , we can recover  $c$  in polynomial time.

*Remark 1.* The above property can also be defined for the  $n$ -parallel 1-out-2 oblivious transfer protocol. If a  $n$ -parallel 1-out-2 oblivious transfer protocol, denoted by  $\text{OT}_n$ , is composed of  $n$  parallel copies of  $\text{OT}$  and if  $\text{OT}$  satisfies uniqueness of transcript property then so does  $\text{OT}_n$ . In particular, given the randomness of the receiver of  $\text{OT}_n$ , it is possible to recover the  $n$  bit length string of the receiver efficiently.

*Instantiation: Naor-Pinkas Protocol [35].* Naor-Pinkas proposed a two message oblivious transfer protocol whose security is based on the Decisional Diffie-Hellman (DDH) assumption.

We claim that their protocol satisfies uniqueness of transcript property. In order to do that, we recall the first message (sent by receiver to sender) in their protocol: Let  $\text{bit}$  be the input of receiver. Consider a group  $\mathbb{G}$  where DDH is hard. Let  $g$  be a generator of  $\mathbb{G}$ . The receiver generates  $g^a, g^b$  and  $c_{\text{bit}} = ab$ . It



generates  $c_{1\text{-bit}}$  at random such that  $c_{\text{bit}} \neq c_{1\text{-bit}}$ . It sends  $v_1 = g^a, v_2 = g^b, v_3 = g^{c_0}, v_4 = g^{c_1}$  to the sender.

The elements  $v_1$  and  $v_2$  uniquely determine  $a$  and  $b$ . Furthermore, exactly one of  $v_3$  or  $v_4$  corresponds to  $g^{ab}$  and this uniquely determines the bit. Furthermore, note that this also uniquely determines the randomness used.

While we only deal with 1-out-2 OT protocol above, we can generalize the above proof to also work for  $n$ -parallel 1-out-2 OT protocol.

**Theorem 4 ([35]).** *Assuming DDH, there exists an oblivious transfer protocol satisfying Definition 5 as well as the uniqueness of transcript property.*

### 3.3 Two Message Secure Function Evaluation

As a building block in our construction, we consider a two message secure function evaluation protocol. Since we are restricted to just two messages, we can only expect one of the parties to get the output.

We designate  $P_1$  to be the party receiving the output and the other party to be  $P_2$ . That is, the protocol proceeds by  $P_1$  sending the first message to  $P_2$  and the second message is the response by  $P_2$ .

*Indistinguishability Security.* We require malicious (indistinguishability) security against  $P_1$  and malicious (indistinguishability) security against  $P_2$ . We define both of them below.

First, we define an indistinguishability security notion against malicious  $P_1$ . To do that, we employ an extraction mechanism to extract  $P_1$ 's input  $x_1^*$ . We then argue that  $P_1$  should not be able to distinguish whether  $P_2$  uses  $x_2^0$  or  $x_2^1$  in the protocol as long as  $f(x_1^*, x_2^0) = f(x_1^*, x_2^1)$ . We don't place any requirements on the computational complexity of the extraction mechanism.

**Definition 6 (Indistinguishability Security: Malicious  $P_1$ ).** *Consider a two message secure function evaluation protocol for a functionality  $f$  between parties  $P_1$  and  $P_2$  such that  $P_1$  is getting the output. We say that the two party secure computation protocol satisfies **indistinguishability security against malicious  $P_1$**  if for every adversarial  $P_1^*$ , there is an extractor  $\text{Ext}$  (not necessarily efficient) such the following holds. Consider the following experiment:*

$\text{Expt}(1^\lambda, b)$ :

- $P_1^*$  outputs the first message  $\text{msg}_1$ .
- Extractor  $\text{Ext}$  on input  $\text{msg}_1$  outputs  $x_1^*$ .
- Let  $x_2^0, x_2^1$  be two inputs such that  $f(x_1^*, x_2^0) = f(x_1^*, x_2^1)$ . Party  $P_2$  on input  $\text{msg}_1$  and  $x_2^b$ , outputs the second message  $\text{msg}_2$ .
- $P_1^*$  upon receiving the second message outputs a bit  $\text{out}$ .
- Output  $\text{out}$ .

We require that,

$$|\Pr[1 \leftarrow \text{Expt}(1^\lambda, 0)] - \Pr[1 \leftarrow \text{Expt}(1^\lambda, 1)]| \leq \text{negl}(\lambda),$$

for some negligible function  $\text{negl}$ .

We now define security against malicious  $P_2$ . We insist that  $P_2$  should not be able to distinguish which input  $P_1$  used to compute its messages.

**Definition 7 (Indistinguishability Security: Malicious  $P_2$ ).** Consider a two message secure function evaluation protocol for a functionality  $f$  between parties  $P_1$  and  $P_2$  where  $P_1$  gets the output. We say that the two party secure computation protocol satisfies **indistinguishability security against malicious  $P_2$**  if for every adversarial  $P_2^*$ , the following holds: Consider two strings  $x_1^0$  and  $x_1^1$ . Denote by  $\mathcal{D}_b$  the distribution of the first message (sent to  $P_2$ ) generated using  $x_1^b$  as  $P_1$ 's input. The distributions  $\mathcal{D}_0$  and  $\mathcal{D}_1$  are computationally indistinguishable.

*Instantiation.* We can instantiate such a two message secure evaluation protocol using garbled circuits and  $\ell_1$ -parallel 1-out-2 two message oblivious transfer protocol OT by Naor-Pinkas [35]. Recall that this protocol satisfies uniqueness of transcript property (Definition 5). We denote the garbling schemes by GC.

We describe this protocol below. The input of  $P_1$  is  $x_1$  and the input of  $P_2$  is  $x_2$ . Recall that  $P_1$  is designated to receive the output.

- $P_1 \rightarrow P_2$ :  $P_1$  computes the first message of OT as a function of its input  $x_1$  of input length  $\ell_1$ . Denote this message by  $OT_1$ . It sends  $OT_1$  to  $P_2$ .
- $P_2 \rightarrow P_1$ :  $P_2$  computes the following:
  - It generates  $\text{Gen}(1^\lambda)$  to get  $\text{gcsk}$ .
  - It then computes  $\text{GrbC}(\text{gcsk}, C)$  to obtain  $\widehat{C}$ .  $C$  is a circuit with  $x_2$  hard-wired in it; it takes as input  $x_1$  and computes  $f(x_1, x_2)$ .
  - It computes  $\text{Grbl}(\text{gcsk})$  to obtain the wire keys  $(\mathbf{k}_1, \dots, \mathbf{k}_{\ell_1})$ , where every  $\mathbf{k}_i$  is composed of two keys  $(k_i^0, k_i^1)$ .
  - It computes the second message of OT, denoted by  $OT_2$ , as a function of  $(\mathbf{k}_1, \dots, \mathbf{k}_{\ell_1})$ .

It sends  $(\widehat{C}, OT_2)$  to  $P_1$ .

- $P_1$ : Upon receiving  $(\widehat{C}, OT_2)$ , it recovers the wire keys  $(k_1, \dots, k_{\ell_1})$ . It then executes  $\text{EvalGC}(\widehat{C}, (k_1, \dots, k_{\ell_1}))$  to obtain **out**. It outputs **out**.

The correctness of the above protocol immediately follows from the correctness of garbling schemes and oblivious transfer protocol. We now focus on security.

**Theorem 5.** Assuming the security of GC and OT and assuming that OT satisfies uniqueness of transcript property (Definition 5), the above protocol is secure against malicious  $P_1$  (Definition 6).

*Proof.* We first describe the inefficient extractor Ext that extracts  $P_1$ 's input from its first message. From the uniqueness of transcript property of OT, it follows that given  $P_1$ 's first message  $OT_1$ , there exists a unique input  $x_1^*$  and randomness  $r$  that was used to compute the message of  $P_1$ . Thus, Ext can find this input  $x_1^*$  by performing a brute force search on all possible inputs and randomness.

We prove the theorem with respect to the extractor described above. In the first hybrid described below, challenge bit  $b$  is used to determine which of the

two inputs of  $P_2$  needs to be picked. In the final hybrid,  $P_2$  always picks the first of the two inputs.

Hyb<sub>1,b</sub> for  $b \stackrel{\$}{\leftarrow} \{0, 1\}$ : Let  $x_1^*$  be the input extracted by the extractor. Let  $x_2^0$  and  $x_2^1$  be two inputs such that  $f(x_1^*, x_2^0) = f(x_1^*, x_2^1)$ . Party  $P_2$  uses  $x_2^b$  to compute the second message.

Hyb<sub>2,b</sub> for  $b \stackrel{\$}{\leftarrow} \{0, 1\}$ : Let  $x_1^*$  be the input extracted by the extractor. We denote the  $i^{\text{th}}$  bit of  $x_1^*$  to be  $x_{1,i}^*$ . As part of the second message, the wire keys  $(\mathbf{k}_1, \dots, \mathbf{k}_{\ell_1})$ , where every  $\mathbf{k}_i$  is composed of two keys  $(k_i^0, k_i^1)$ . Instead of generating  $OT_2$  as a function of  $(\mathbf{k}_1, \dots, \mathbf{k}_{\ell_1})$ , it generates  $OT_2$  as a function of  $(\mathbf{k}'_1, \dots, \mathbf{k}'_{\ell_1})$ .  $\mathbf{k}'_i$  contains  $(0, k_i^{x_{1,i}^*})$  if  $x_{1,i}^* = 1$ , otherwise it contains  $(k_i^{x_{1,i}^*}, 0)$ .

Hybrids Hyb<sub>1,b</sub> and Hyb<sub>2,b</sub> are computationally distinguishable from the indistinguishability security against malicious receivers property of the oblivious transfer protocol.

Hyb<sub>3,0</sub>: Let  $x_1^*$  be the input extracted by the extractor. Let  $x_2^0$  and  $x_2^1$  be two inputs such that  $f(x_1^*, x_2^0) = f(x_1^*, x_2^1)$ .  $P_2$  computes the second message as in the previous hybrid. Instead of using  $x_2^b$  in the computation of the garbled circuit, it instead uses the input  $x_2^0$ .

Hybrids Hyb<sub>2,b</sub> and Hyb<sub>3,0</sub> are computationally indistinguishable from the security of the garbling schemes<sup>11</sup>.

The final hybrid does not contain any information about the challenge bit. This completes the proof.

**Theorem 6.** *Assuming the security of OT, the above protocol is secure against malicious  $P_2$  (Definition 7).*

*Proof.* The proof of this theorem directly follows from the security against malicious senders property of the oblivious transfer protocol.

### 3.4 Conditional Disclosure of Secrets (CDS) Protocols

We require another key primitive, conditional disclosure of secrets (CDS) [19,1] protocol. A CDS protocol consists of two parties  $P_1$  and  $P_2$ . Both these parties share a common instance  $\mathbf{X}$  belonging to a NP language. Further,  $P_2$  has a secret  $s$  and  $P_1$  additionally has a private input  $w$ . If  $w$  is a valid witness for  $\mathbf{X}$  then we require that  $P_1$  should be able to recover the secret  $s$  at the end of the protocol. However, if  $\mathbf{X}$  does not belong to the language then we require that  $P_1$  does not get any information about the secret.

We give the formal definition below.

<sup>11</sup> Formally this is argued by first simulating the garbled circuit and then switching the input.

**Definition 8 (CDS Protocols).** *Conditional Disclosure of Secret protocol, associated with a NP relation  $\mathcal{R}$ , is an interactive protocol between two parties  $P_1$  (receiver) and  $P_2$  (sender). Both  $P_1$  and  $P_2$  hold the same instance  $\mathbf{X}$ . Party  $P_2$  holds the secret  $s \in \{0, 1\}^\lambda$  and  $P_1$  holds a string  $w \in \{0, 1\}^*$ . At the end of the protocol  $P_1$  outputs  $s'$ . We denote this by  $s' \leftarrow \langle P_1(\mathbf{X}, w), P_2(\mathbf{X}, s) \rangle$ .*

*We require that the CDS protocol satisfies the following properties:*

- **Correctness:** *If  $(\mathbf{X}, w) \in \mathcal{R}$  then it holds with probability 1 that  $s \leftarrow \langle P_1(\mathbf{X}, w), P_2(\mathbf{X}, s) \rangle$ .*
- **Soundness:** *If  $\mathbf{X} \notin L(\mathcal{R})$  then, for any boolean distinguisher  $P_1^*$ , for any  $s_0, s_1 \in \{0, 1\}^\lambda$  and for any auxiliary information  $\mathbf{z} \in \{0, 1\}^*$ , it holds that,*

$$\begin{aligned} & |\Pr[1 \leftarrow \langle P_1^*(\mathbf{X}, s_0, s_1, \mathbf{z}), P_2(\mathbf{X}, s_0) \rangle] - \Pr[1 \leftarrow \langle P_1^*(\mathbf{X}, s_0, s_1, \mathbf{z}), P_2^*(\mathbf{X}, s_1) \rangle]| \\ & \leq \text{negl}(\lambda) \end{aligned}$$

*for some negligible function  $\text{negl}$ .*

*Construction of Two Message CDS protocol.* Since a CDS protocol is a special case of two party secure computation, we show how a two message secure function evaluation protocol (Section 3.3) implies a two message CDS protocol.

**Theorem 7.** *Consider a NP relation  $\mathcal{R}$ . Consider the following two party functionality  $f$  that takes as input  $((\mathbf{X}', w); (\mathbf{X}, s))$  and outputs  $s$  if and only if  $((\mathbf{X}, w) \in \mathcal{R}) \wedge \mathbf{X} = \mathbf{X}'$ , otherwise it outputs 0. A two message secure function evaluation protocol for  $f$  is a CDS protocol associated with the relation  $\mathcal{R}$ .*

*Proof.* The correctness of the CDS protocol immediately follows from the correctness of the two message secure function evaluation protocol. We now argue soundness.

Consider an instance  $\mathbf{X} \notin \mathcal{L}(\mathcal{R})$ . We now invoke the security of two message SFE (specifically, Definition 6). There exists an extractor  $\text{Ext}$  that extracts  $x_1^*$  from  $P_1^*$ 's first message. We claim that for every  $x_2$  of the form  $(\mathbf{X}, s')$ , it holds that  $f(x_1^*, x_2)$  outputs 0. This follows from the fact that  $\mathbf{X} \notin \mathcal{L}(\mathcal{R})$ . Using this fact, it follows that  $P_1^*$  cannot distinguish whether  $P_2$  used the input  $(\mathbf{X}, s_0)$  or  $(\mathbf{X}, s_1)$  to compute its message. The theorem thus follows.

### 3.5 Zero Knowledge Proof Systems

We now recall the notion of zero knowledge [23]. In the definition below, we consider computationally bounded provers.

**Definition 9 (Zero Knowledge Argument of Knowledge).** *A Zero Knowledge Argument of Knowledge (ZKAoK) system (Prover, Verifier) for a relation  $\mathcal{R}$ , associated with a NP language  $\mathcal{L}(\mathcal{R})$ , is an interactive protocol between Prover and Verifier. Prover takes as input  $(\mathbf{y}, \mathbf{w})$  and verifier Verifier takes as input  $\mathbf{y}$ . At the end of the protocol, verifier outputs accept/reject. This process is denoted by  $\langle \text{Prover}(\mathbf{y}, \mathbf{w}), \text{Verifier}(\mathbf{y}) \rangle$ . It consists of the following properties:*

- **Completeness:** For every  $(\mathbf{y}, \mathbf{w}) \in \mathcal{R}$ , we have:

$$\Pr[\text{accept} \leftarrow \langle \text{Prover}(\mathbf{y}, \mathbf{w}), \text{Verifier}(\mathbf{y}) \rangle] = 1$$

- **Extractability:** For every PPT  $\text{Prover}^*$ , there exists an extractor  $\text{Ext}$  (that could use the code of  $\text{Prover}^*$  in a non black box manner) such that the following holds: for every auxiliary information  $z \in \{0, 1\}^*$ ,

$$\left| \Pr[\text{accept} \leftarrow \langle \text{Prover}^*(\mathbf{y}, z), \text{Verifier}(\mathbf{y}) \rangle] - \Pr[\mathbf{w}^* \leftarrow \text{Ext}(1^\lambda, z) : (\mathbf{y}, \mathbf{w}^*) \in \mathcal{R}] \right| \leq \text{negl}(\lambda)$$

- **Zero Knowledge:** For every  $(\mathbf{y}, \mathbf{w}) \in \mathcal{R}$ , for every PPT  $\text{Verifier}^*$ , there exists a PPT simulator  $\text{Sim}$  (that could use the code of  $\text{Verifier}^*$  in a non black box manner) such that the following holds:

$$\{\langle \text{Prover}(\mathbf{y}, \mathbf{w}), \text{Verifier}^*(\mathbf{y}) \rangle\} \approx_c \{\text{Sim}(1^\lambda, \mathbf{y})\}$$

We define a ZKAoK system to be  $k$ -message if the number of messages between Prover and Verifier is  $k$ .

We require zero knowledge systems satisfying additional properties. We consider them one by one.

*Bounded Uniform Zero Knowledge.* In the zero knowledge property considered in the definition above, we require that the malicious verifier is uniform.

**Definition 10 ( $\mu$ -Bounded Uniform Zero Knowledge).** A proof system  $(\text{Prover}, \text{Verifier})$  for a relation  $\mathcal{R}$  is said to be  $\mu$ -bounded uniform ZKAoK if the following holds:

- It satisfies correctness and extractability properties as in Definition 9.
- **$\mu$ -Bounded Uniform Zero Knowledge:** For every  $(\mathbf{y}, \mathbf{w}) \in \mathcal{R}$ , for every PPT  $\text{Verifier}^*$  (represented as a Turing machine), there exists a PPT simulator  $\text{Sim}$  (that could use the code of  $\text{Verifier}^*$  in a non black box manner) such that the following holds: for any auxiliary information  $\mathbf{z} \in \{0, 1\}^{\mu(|\mathbf{y}|)}$ .

$$\{\langle \text{Prover}(\mathbf{y}, \mathbf{w}), \text{Verifier}^*(\mathbf{y}, \mathbf{z}) \rangle\} \approx_c \{\text{Sim}(1^\lambda, \mathbf{y}, \mathbf{z})\}$$

*Remark 2.* The special case of 0-bounded uniform zero knowledge (interpreted as a constant function that always outputs 0) reduces to having the malicious verifiers as uniform algorithms (in particular, they receive no external advice).

*Delayed Statement-Witness.* Another useful property we require is to be able to choose the statement and the witness in the last message of the protocol. We call this, delayed statement-witness property.

**Definition 11 (Delayed Statement-Witness).** A Zero Knowledge (proof or argument) system is said to satisfy delayed statement-witness property if both the statement and the witness are fixed only in the last message of the protocol. In particular, all the messages except the last message depend only on the length of the instance and the witness.

*Instantiation.* In this work, we require a ZKAoK system that is both bounded uniform zero knowledge and satisfies delayed statement-witness property. The protocol of Bitansky et al. [8] satisfies both these properties. Their protocol can be instantiated from Zaps [14], DDH and the Learning with Errors (LWE) assumption.

**Theorem 8 ([8]).** *Assuming Zaps, DDH and LWE, there exists a ZKAoK system that satisfies both  $\mu$ -bounded uniform zero knowledge for some function  $\mu$ , and delayed statement-witness property.*

### 3.6 Succinct Randomized Encodings

We recall the notion of succinct randomized encodings [9,12,32] next.

**Definition 12.** *A succinct randomized encodings scheme  $SRE = (E, D)$  for a class of Turing machines  $\mathcal{M}$  consists of the following probabilistic polynomial time algorithms:*

- **Encoding,**  $E(1^\lambda, M, x)$ : *On input security parameter  $\lambda$ , Turing machine  $M \in \mathcal{M}$  and input  $x$ , it outputs the randomized encoding  $\langle M, x \rangle$ .*
- **Decoding,**  $D(\langle M, x \rangle)$ : *On input randomized encoding of  $M$  and  $x$ , it outputs **out**.*

*We require that the above algorithms satisfies the following properties:*

- **Correctness:** *We require that the following holds for every  $M \in \mathcal{M}, x \in \{0, 1\}^*$ ,*

$$\Pr [D(\langle M, x \rangle) = M(x) : \langle M, x \rangle \leftarrow E(1^\lambda, M, x)] = 1$$

- **Security:** *For every PPT adversary  $\mathcal{A}$ , there exists a PPT simulator  $\text{Sim}$  such that the following holds:*

$$\{\langle M, x \rangle\} \approx_c \left\{ \text{Sim}(1^\lambda, 1^{|M|}, 1^{|x|}, M(x)) \right\},$$

*where:*

- $\langle M, x \rangle \leftarrow E(1^\lambda, M, x)$

*Input-less Turing machines.* In this work, we consider input-less Turing machines. These are Turing machines which on input  $\perp$ , executes some computation and outputs **out**. We denote the randomized encoding of an input-less TM to be  $\langle M \rangle \leftarrow E(1^\lambda, M, \perp)$ .

### 3.7 Indistinguishability Obfuscation for Circuits

We define the notion of indistinguishability obfuscation (iO) for circuits [4,17] below.

**Definition 13 (Indistinguishability Obfuscator (iO) for Circuits).** A uniform PPT algorithm  $\text{iO}$  is called an  $\varepsilon$ -secure indistinguishability obfuscator for a circuit family  $\{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ , where  $\mathcal{C}_\lambda$  consists of circuits  $C$  of the form  $C : \{0, 1\}^\ell \rightarrow \{0, 1\}$ , if the following holds:

- **Completeness:** For every  $\lambda \in \mathbb{N}$ , every  $C \in \mathcal{C}_\lambda$ , every input  $x \in \{0, 1\}^\ell$ , where  $\ell = \ell(\lambda)$  is the input length of  $C$ , we have that

$$\Pr[C'(x) = C(x) : C' \leftarrow \text{iO}(\lambda, C)] = 1$$

- **$\varepsilon$ -Indistinguishability:** For any PPT distinguisher  $D$ , there exists a negligible function  $\text{negl}(\cdot)$  such that the following holds: for all sufficiently large  $\lambda \in \mathbb{N}$ , for all pairs of circuits  $C_0, C_1 \in \mathcal{C}_\lambda$  such that  $C_0(x) = C_1(x)$  for all inputs  $x \in \{0, 1\}^\ell$ , where  $\ell = \ell(\lambda)$  is the input length of  $C_0, C_1$ , we have:

$$\left| \Pr[D(\lambda, \text{iO}(\lambda, C_0)) = 1] - \Pr[D(\lambda, \text{iO}(\lambda, C_1)) = 1] \right| \leq \varepsilon$$

If  $\varepsilon$  is negligible in  $\lambda$  then we refer to  $\text{iO}$  as a secure indistinguishability obfuscator.

*Remark 3.* In our work, we require indistinguishability obfuscators where the indistinguishability property holds against adversaries running in sub-exponential time (rather than polynomial time). We refer to such indistinguishability obfuscators as sub-exponentially secure indistinguishability obfuscators. Currently, the existence of several cryptographic primitives are based only on the assumption of sub-exponential iO.

### 3.8 Puncturable Pseudorandom Functions

We define the notion of puncturable pseudorandom functions below.

**Definition 14.** A pseudorandom function of the form  $\text{PRF}_{\text{punc}}(K, \cdot)$  is said to be a  $\mu$ -secure puncturable PRF if there exists a PPT algorithm  $\text{Puncture}$  that satisfies the following properties:

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

$$\left| \Pr[\mathcal{A}_2(K \setminus \{x\}, x, \text{PRF}_{\text{punc}}(K, x)) = 1] - \Pr[\mathcal{A}_2(K \setminus \{x\}, x, U_{\chi(\lambda)}) = 1] \right| \leq \mu(\lambda)$$

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

If  $\mu$  is negligible, we refer to  $\text{PRF}_{\text{punc}}$  as a secure puncturable PRF.



As observed by [10,11,31], the GGM construction [20] of PRFs from one-way functions yields puncturable PRFs.

**Theorem 9 ([20,10,11,31]).** *If  $\frac{\mu}{\text{poly}}$ -secure one-way functions exist, for some fixed polynomial poly, then there exists  $\mu$ -secure puncturable pseudorandom functions.*

## 4 Generation Protocols

A crucial ingredient in our two party secure computation protocol is a protocol that enables extraction of the input of  $P_2$  during the simulation phase. To achieve this, we introduce the notion of generation protocols<sup>12</sup> below.

This is a two party protocol between a sender and a receiver. The sender has a trapdoor and in the end of the protocol, the receiver outputs a string. It consists of two properties: (i) *soundness*: any adversarial receiver having black-box access to the code of the sender will not be able to recover the trapdoor of the sender, (ii) *extractability*: an extractor can successfully recover the trapdoor of the sender. In the extractability property, we only consider the case when the sender is semi-honest (i.e., it behaves according to the description of the protocol).

To make sure that both soundness and extractability don't contradict each other, we make sure that the extractor has more capabilities than an adversarial receiver – for instance, an extractor could rewind the receiver or it could have non black box access to the code of the receiver.

The formal definition of generation protocols is provided below.

**Definition 15 (Generation Protocols).** *A generation protocol is an interactive protocol between two parties  $P_1$  (also termed receiver) and  $P_2$  (also termed sender). The input to both parties is auxiliary information  $\mathbf{z}$ . Party  $P_2$ , in addition, gets as input trapdoor  $K \in \{0, 1\}^{\text{poly}(\lambda)}$ . At the end of the protocol,  $P_1$  outputs  $K'$ . We denote this process by  $K' = \langle P_1(\mathbf{z}), P_2(\mathbf{z}, K) \rangle$ .*

*The following properties are associated with a generation protocol:*

- **Soundness:** *For any PPT non-uniform boolean distinguisher  $P_1^*$ , for any large enough security parameter  $\lambda \in \mathbb{N}$ : for every two strings  $K_0, K_1 \in \{0, 1\}^{\text{poly}(\lambda)}$  and auxiliary information  $\mathbf{z} \in \{0, 1\}^{\text{poly}'(\lambda)}$ ,*

$$\begin{aligned} & |\Pr[1 \leftarrow \langle P_1^*(\mathbf{z}, K_0, K_1), P_2(\mathbf{z}, K_0) \rangle] - \Pr[1 \leftarrow \langle P_1^*(\mathbf{z}, K_0, K_1), P_2(\mathbf{z}, K_1) \rangle]| \\ & \leq \frac{1}{2} + \text{negl}(\lambda) \end{aligned}$$

*for some negligible function negl. That is, any distinguisher  $P_1^*$  having black box access to  $P_2$  cannot distinguish whether which of  $K_0$  and  $K_1$  was used in the protocol.*

<sup>12</sup> The name “generation protocol” is taken from the work of [3]. The definition in their work is slightly different, however they too use the notion of generation protocols to achieve trapdoor extraction.

- **Extractability:** For every semi-honest PPT  $P_2^*$ , there exists a PPT extractor  $\text{ExtGP}$  (that could possibly use code of  $P_2^*$  in a non black box manner) such that the following holds: for any auxiliary information  $\mathbf{z} \in \{0, 1\}^{\text{poly}'(\lambda)}$ ,
  - The view of  $P_2^*(\mathbf{z}, K)$  when it is interacting with  $P_1(\mathbf{z}, K)$  is computationally indistinguishable from the view of  $P_2^*(\mathbf{z}, K)$  when it is interacting with  $\text{ExtGP}(1^\lambda, \mathbf{z})$ .
  - $\Pr [K' \leftarrow \langle \text{ExtGP}(1^\lambda, \mathbf{z}), P_2^*(\mathbf{z}, K) \rangle \text{ and } K' = K] \geq 1 - \text{negl}(\lambda)$

*Extractability Against  $\mu$ -Bounded Uniform Senders.* We consider generation protocols where the extractability property needs to hold against senders modeled as  $\mu$ -bounded uniform algorithms. We formally define this below.

**Definition 16.** A protocol  $\text{GenProt}$  between sender  $P_1$  and receiver  $P_2$  is said to be  $\mu$ -bounded uniform generation protocol if the following holds:

- It satisfies the soundness property in Definition 16.
- **Extractability against  $\mu$ -bounded uniform senders:** For every semi-honest PPT  $P_2$  (modeled as a Turing machine), there exists a PPT extractor  $\text{ExtGP}$  (that could possibly use code of  $P_2$  in a non black box manner) such that the following holds: for any bounded auxiliary information  $\mathbf{z} \in \{0, 1\}^{\mu(\lambda)}$ ,

$$\Pr [K' \leftarrow \langle \text{ExtGP}(1^\lambda, \mathbf{z}), P_2(\mathbf{z}, K) \rangle \text{ and } K' = K] \geq 1 - \text{negl}(\lambda)$$

*Remark 4.* If  $\mu$  in the above definition is a constant function that always outputs 0 then this boils down to the case when the sender is a uniform algorithm (hence, no external advice). In this case, we refer to the above generation protocol as uniform generation protocol.

#### 4.1 Two-Message GP from Succinct RE

We present a two-message generation protocol starting from a succinct randomized encoding scheme and a two party secure function evaluation protocol. The security of this scheme will be against  $\mu$ -bounded uniform senders.

*Tools.* The first tool we use is succinct randomized encodings for Turing machines, denoted by  $\text{SRE} = (\text{E}, \text{D})$ . Another tool we use is a two message secure function evaluation protocol  $\Pi_{2\text{PC}}$ . In particular, we use the two message secure function evaluation protocol defined in Section 3.3. We denote  $\mathcal{P}_1$  and  $\mathcal{P}_2$  to be the parties involved in this protocol. Only  $\mathcal{P}_1$  outputs in the protocol. Recall that this protocol satisfies indistinguishability security (Definitions 6 and 7).

**FUNCTIONALITY OF  $\Pi_{2\text{PC}}$ :** The functionality  $f$  associated with  $\Pi_{2\text{PC}}$  is the following:  $f$  on input  $x_2 = (\beta, K, m, R_2, \text{md}, \theta)$  from  $\mathcal{P}_2$  and  $x_1 = (M, R_1)$  (here,  $|M| \leq \mathcal{O}(\mu(\lambda) + \lambda)$ ) from  $\mathcal{P}_1$ , it computes the following:

- If  $\text{md} = 1$  then compute the succinct randomized encoding  $\langle N \rangle \leftarrow \text{E}(1^\lambda, N[\beta, K, m, M], \perp; R)$  (i.e.,  $R$  is the randomness used in  $\text{E}$ ), where  $R$  is set to  $R_1 \oplus R_2$ . The Turing machine  $N$  is an input-less Turing machine (refer Section 3.6) that does the following: hardwired inside it are the values  $(\beta, K, m, M)$ .

1. It first computes  $M(m)$  to get as output  $\text{out}$ .
2. It interprets the first  $|\beta|$  number of bits of  $\text{out}$  to be the string  $\beta'$ .
3. It checks if  $\beta' = \beta$ . If so, it outputs  $K$ . Otherwise, it outputs  $\perp$ .

It outputs  $\langle N \rangle$ .

– If  $\text{md} = 2$  then:

1. It outputs  $\theta$ .

*Construction.* We describe the protocol below. Denote the receiver to be  $P_1$  and the sender to be  $P_2$ . Call this protocol **GenProt**.

- Upon input  $\mathbf{z}$ ,  $P_1$  (receiver) prepares an input  $x_1$  for  $\Pi_{2\text{PC}}$  as a  $\mu(\lambda)$ -length string of all zeroes. It takes the role of the party  $\mathcal{P}_1$  in the protocol  $\Pi_{2\text{PC}}$ . It computes the first message  $\mathbf{msg}_1$  of  $\Pi_{2\text{PC}}$  using the input  $x_1$ . That is,  $\mathbf{msg}_1 \leftarrow \mathcal{P}_1.\text{MsgGen}[\Pi_{2\text{PC}}](1^\lambda, x_1)$ . It sends  $\mathbf{msg}_1$  to  $P_2$  (sender).
- Upon input  $\mathbf{z}$  and trapdoor  $K$ ,  $P_2$  (sender) first picks a string  $\beta$  of length  $\ell_\beta = \text{poly}(\lambda)$  such that  $\ell_\beta \gg |\mathbf{msg}_1|$ . In particular, we require that  $2^{-(\ell_\beta - \mu(\lambda) - \lambda)}$  to be negligible. It sets  $m = \mathbf{msg}_1$ . It samples a string  $R$  uniformly at random.

It takes the role of  $\mathcal{P}_2$  in the protocol  $\Pi_{2\text{PC}}$ . It then sets its input to  $\Pi_{2\text{PC}}$  to be  $x_2 = (\beta, K, \mathbf{msg}_1, R, \text{md}, \theta)$ , where  $\text{md} = 1$  and  $\theta = 0$ . Using  $x_2$  and  $\mathbf{msg}_1$ , it computes the second message  $\mathbf{msg}_2$  of  $\Pi_{2\text{PC}}$  using the input  $x_2$ . That is,  $\mathbf{msg}_2 \leftarrow \mathcal{P}_2.\text{MsgGen}[\Pi_{2\text{PC}}](1^\lambda, x_2, \mathbf{msg}_1)$ . It sends  $(\beta, \mathbf{msg}_2)$  to  $P_1$ .

Finally,  $P_1$  computes the output of  $\Pi_{2\text{PC}}$  and recovers the randomized encoding  $\langle N \rangle$ . It then evaluates the decoding algorithm  $D(\langle N \rangle)$  to get the output  $K'$ . It outputs  $K'$ .

This concludes the construction. We argue that the above protocol satisfies the properties of the generation protocol.

**Theorem 10.** *Assuming the security of  $\Pi_{2\text{PC}}$  (Definition 7) and SRE, GenProt satisfies soundness.*

*Proof.* Suppose  $P_1^*$  receives as input two trapdoors  $K_0$  and  $K_1$ . In this case we need to argue that a malicious  $P_1^*$  having just black box access to (honest)  $P_2$  will be unable to distinguish whether  $P_2$  is using  $K_0$  or  $K_1$ . In fact, we argue a stronger property: we argue that the behavior of  $P_1^*$  can be simulated by a PPT simulator even without the knowledge of  $K$ . That is, for every adversarial receiver  $P_1^*$ , there exists a PPT simulator  $\text{Sim}$ , for every  $K \in \{0, 1\}^{\text{poly}(\lambda)}$  and auxiliary information  $\mathbf{z} \in \{0, 1\}^{\text{poly}(\lambda)}$ ,

$$|\Pr[1 \leftarrow \langle P_1^*(\mathbf{z}), P_2(\mathbf{z}, K) \rangle] - \Pr[1 \leftarrow \langle P_1^*(\mathbf{z}), \text{Sim}(\mathbf{z}) \rangle]| \leq \frac{1}{2} + \text{negl}(\lambda)$$

Note that the above property implies soundness property.

*Description of Sim(z).* It receives as input  $\text{msg}_1$  from  $P_1$ . It generates  $\text{msg}_2$  as follows:

- Let  $\text{Sim}_{\text{SRE}}$  be the simulator of the succinct randomized encodings scheme. It then executes  $\text{Sim}_{\text{SRE}}(1^\lambda, 1^{\ell_1}, 1^{\ell_2}, v)$ , where  $\ell_1$  is the size of  $M$ ,  $\ell_2$  is the size of  $m$  as defined in the description of functionality for  $\Pi_{2\text{PC}}$  and  $v$  is set to be  $\perp$ . The output of  $\text{Sim}_{\text{SRE}}(1^\lambda, 1^{\ell_1}, 1^{\ell_2}, v)$  is denoted by  $\langle N \rangle$ .
- It sets  $x_2 = (0, 0, 0, 0, 2, \langle N \rangle)$ . It then computes  $\text{msg}_2$  as a function of  $x_2$  and  $\text{msg}_1$ . The generation of  $\text{msg}_2$  is performed by running the algorithm of (honest)  $\mathcal{P}_2$  in  $\Pi_{2\text{PC}}$ . That is,  $\text{msg}_2 \leftarrow \mathcal{P}_2.\text{MsgGen}[\Pi_{2\text{PC}}](1^\lambda, x_2, \text{msg}_1)$ .
- Finally, it samples a string  $\beta$  of length  $\ell_\beta$ .

Sim then sends  $(\beta, \text{msg}_2)$  to  $P_2$ . This ends the description of Sim.

We focus on proving the above stronger property. In the following hybrids, we use extractor Ext associated with  $\Pi_{2\text{PC}}$  (see Definition 6). Recall that Ext need not necessarily be efficient.

**Hyb<sub>1</sub>:** This corresponds to the real experiment where  $P_1^*(z)$  is interacting with  $P_2(z, K)$ . The output of this hybrid is the output of  $P_1^*$ .

**Hyb<sub>2</sub>:** In this hybrid, party  $P_2$  deviates from the description of the protocol. It uses the extractor Ext to extract  $x_1^* = (M, R_1)$ . It then sets  $x_2' = (0, 0, 0, 0, 2, \theta)$  and uses this input to generate the second message of the protocol  $\Pi_{2\text{PC}}$ . That is,  $\text{msg}_2 \leftarrow \mathcal{P}_2.\text{MsgGen}[\Pi_{2\text{PC}}](1^\lambda, x_2', \text{msg}_1)$ , where  $\text{msg}_1$  is the message sent by  $P_1^*$ . Here,  $\theta$  is set to be the output  $f(x_1^*, (\beta, K, \text{msg}_1, R_2, 1, 0))$ , where  $R_2$  is sampled uniformly at random.  $P_2$  sends  $(\beta, \text{msg}_2)$  to  $P_1^*$ , where  $\beta$  is a string of length  $\ell_\beta$  sampled uniformly at random. The output of this hybrid is the output of  $P_1^*$ .

Since Ext need not be efficient,  $P_2$  is not necessarily efficient.

*Claim.* Assuming the security of  $\Pi_{2\text{PC}}$ , hybrids Hyb<sub>1</sub> and Hyb<sub>2</sub> are computationally indistinguishable.

*Proof.* Suppose  $x_1^* = (M, R_1)$ , interpreted as the description of a Turing machine  $M$  (with the bounded auxiliary information part of this) along with randomness  $R_1$ , is the input extracted by the extractor Ext from the first message of the generation protocol. Let  $x_2'$  be the input used by  $\mathcal{P}_2$  in Hyb<sub>1</sub> and let  $x_2''$  be the input used by  $\mathcal{P}_2$  in Hyb<sub>2</sub>. We have that  $f(x_1, x_2') = f(x_1, x_2'')$ . And thus, from the security of  $\Pi_{2\text{PC}}$  (Definition 6), we have that  $P_1^*$  cannot distinguish whether  $P_2$  used  $x_2'$  or  $x_2''$ . The claim thus follows.

**Hyb<sub>3</sub>:** In this hybrid,  $P_2$  essentially executes the simulator Sim described above.

*Claim.* Assuming the security of SRE, hybrids Hyb<sub>2</sub> and Hyb<sub>3</sub> are computationally indistinguishable.

*Proof.* Suppose  $x_1^* = (M, R_1)$ , interpreted as a Turing machine  $M$  (with the auxiliary information hardcoded in it) along with randomness  $R_1$ , is the input

extracted by the extractor  $\text{Ext}$  from the first message  $\text{msg}_1$  of  $\Pi_{2\text{PC}}$ . Sample string  $\beta$  of length  $\ell_\beta$  uniformly at random. We first make the following observation. The probability that for any  $\gamma$ ,  $M(\gamma)$  outputs the random string  $\beta$  is at most  $2^{-\mathcal{O}(\ell_\beta - \mu(\lambda) - \lambda)}$ , which is negligible. Thus with overwhelming probability we have that  $N[\beta, K, \text{msg}_1, M]$  outputs  $\perp$ .

The only difference between  $\text{Hyb}_2$  and  $\text{Hyb}_3$  is that in  $\text{Hyb}_2$ ,  $\theta$  is set to  $\langle N \rangle$  whereas in  $\text{Hyb}_3$ ,  $\theta$  is set to be the simulated randomized encoding corresponding to the output  $\perp$ . As observed above,  $N$  outputs  $\perp$  except with negligible probability. Thus, we can invoke the security of randomized encodings to argue that  $\text{Hyb}_2$  and  $\text{Hyb}_3$  are computationally indistinguishable.

From the indistinguishability of  $\text{Hyb}_1$  and  $\text{Hyb}_3$ , we have that  $P_1^*$  cannot distinguish whether it is interacting with  $P_2$  versus interacting with  $\text{Sim}$ . This completes the proof.

**Theorem 11.** *Assuming the correctness, security properties of  $\Pi_{2\text{PC}}$  (Definition 6) and SRE,  $\text{GenProt}$  satisfies extractability against  $\mu$ -uniform senders.*

*Proof.* We design an extractor  $\text{ExtGP}$  that extracts the trapdoor from the semi-honest sender  $P_2^*$ . The extractor has the knowledge of the code used by  $P_2^*$ . Call the Turing machine executed by  $P_2^*$  to be  $M$  (which has auxiliary information hardcoded in it). Since we are assuming that  $P_2^*$  is  $\mu$ -bounded uniform, we have  $|M| \leq \mathcal{O}(\mu(\lambda) + \lambda)$ : this is to account for the auxiliary information whose length is at most  $\mu(\lambda)$  and representing the Turing machine requires size at most  $\lambda$ .

Now, the extractor proceeds as follows: it sets the input to  $\Pi_{2\text{PC}}$  to be  $M$ . It then computes the first message  $\text{msg}_1$  of  $\Pi_{2\text{PC}}$  and sends it to  $P_2^*$ . Then,  $P_2^*$  computes  $(\beta, \text{msg}_2)$  and sends it to the extractor.

- From the security of  $\Pi_{2\text{PC}}$  (Definition 6), the view of  $P_2^*$  when interacting with  $P_1$  is computationally indistinguishable from the view of  $P_2$  when interacting with  $\text{ExtGP}$ . Recall that  $P_1$  uses the input 0 in the first message and  $\text{ExtGP}$  uses the input  $M$  in the first message.
- Since  $P_2^*$  is semi-honest, it computes the second message of  $\Pi_{2\text{PC}}$  honestly. From the correctness of  $\Pi_{2\text{PC}}$ , it follows that the extractor can recover the randomized encoding  $\langle N \rangle$  from  $\Pi_{2\text{PC}}$ . From the correctness of SRE, it further follows that the decoding of  $\langle N \rangle$  yields  $K$  if and only if the first  $\ell_\beta$  bits of  $M(\text{msg}_1)$  yields  $\beta$ . Since  $M$  was chosen to be the code of  $P_2^*$ , it follows that the decoding of  $\langle N \rangle$  does yield  $K$ .

From the above two bullets, we have that  $\text{GenProt}$  satisfies extractability property.

## 5 Three-Round Secure Computation

Consider any boolean functionality  $f : \{0, 1\}^{\ell_1} \times \{0, 1\}^{\ell_2} \rightarrow \{0, 1\}$ , where the output is delivered to the second party. We construct a three-round secure two-party computation protocol  $\Pi_{2\text{PC}}$  that securely computes  $f$  against bounded non-uniform adversaries. We denote the two parties involved in the protocol as  $P_1$  and  $P_2$ .

*Building Blocks.* We describe the building blocks used in our protocol.

1. GARBLING SCHEME FOR CIRCUITS (Definition 3), denoted by  $\text{GC} = (\text{Gen}, \text{GrbC}, \text{Grbl}, \text{EvalGC})$ . Without loss of generality we can assume that  $\text{GrbC}$  and  $\text{Grbl}$  are deterministic algorithms.

2. TWO MESSAGE  $\ell_2$ -PARALLEL 1-OUT-2 OBLIVIOUS TRANSFER PROTOCOL (Definition 4), denoted by  $\text{OT}$ . We require security against malicious receivers. We additionally require that the  $\text{OT}$  protocol satisfies uniqueness of transcript property (Definition 5).

3. THREE MESSAGE ZERO KNOWLEDGE ARGUMENT OF KNOWLEDGE (ZKAoK) SYSTEM (Definition 9) for NP. We require that the 3-message ZKAoK system  $\text{ZK} = (\text{Prover}, \text{Verifier})$  satisfies the delayed statement-witness property (Definition 11).

We denote the relation associated with the above system to be  $\mathcal{R}_{\text{zk}}$ . And let  $\mathcal{L}(\mathcal{R}_{\text{zk}})$  be the associated language. The relation  $\mathcal{R}_{\text{zk}}$  is described in Figure 2.

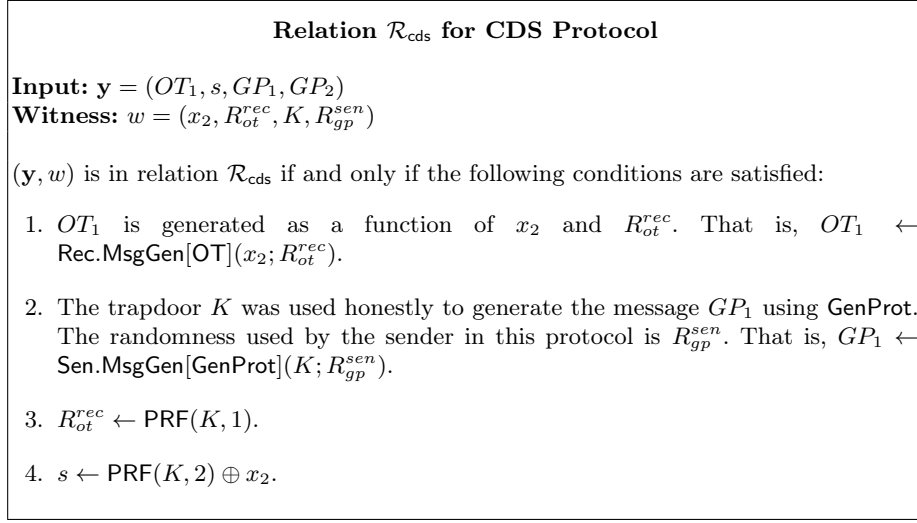
4. TWO MESSAGE GENERATION PROTOCOL (Definition 16) denoted by  $\text{GenProt}$ . In particular, we are interested in generation protocols satisfying *special extraction* property. We consider a two message generation protocol. The role of the sender of  $\text{GenProt}$  is played by  $P_2$  and the role of the receiver of  $\text{GenProt}$  is played by  $P_1$ .

5. TWO MESSAGE CONDITIONAL DISCLOSE OF SECRET (CDS) PROTOCOL (Definition 8), denoted by  $\text{CDSProt}$ . The associated relation  $\mathcal{R}_{\text{cds}}$  is described in Figure 1.

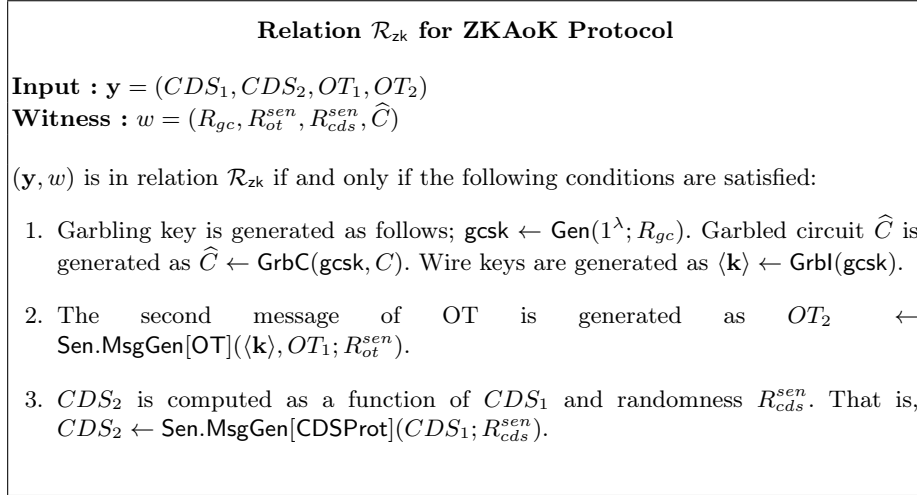
6. OTHER TOOLS. We additionally use pseudorandom functions, denoted by  $\text{PRF}$ , in this construction.

*Protocol  $\Pi_{2\text{PC}}$ .* We now proceed to describe protocol  $\Pi_{2\text{PC}}$ .

1.  $P_1 \rightarrow P_2$ : On input  $x_1$  of length  $\ell_1$ , party  $P_1$  does the following:
  - Compute the prover’s message of ZK, denoted by  $ZK_1$ .
  - It computes the first message of the generation protocol using randomness  $R_{gp}^{\text{rec}}$ . That is,  $GP_1 \leftarrow \text{Rec.MsgGen}[\text{GenProt}](R_{gp}^{\text{rec}})$ . It sends  $(ZK_1, GP_1)$  to  $P_2$ .
2.  $P_2 \rightarrow P_1$ : Party  $P_2$  computes the third message as follows:
  - Compute the verifier’s message of ZK. Denote this by  $ZK_2$ .
  - It computes  $R_{ot}^{\text{rec}} = \text{PRF}(K, 1)$ , randomness used in  $\text{OT}$ .
  - It computes the first message of  $\text{OT}$ , denoted by  $OT_1$ , as a function of its input  $x_2$  and randomness  $R_{ot}^{\text{rec}}$ . That is,  $OT_1 \leftarrow \text{Rec.MsgGen}[\text{OT}](x_2; R_{ot}^{\text{rec}})$ , where  $\text{Rec}$  is the receiver algorithm of  $\text{OT}$ . Here,  $x_2$  is interpreted as a vector with the  $i^{\text{th}}$  entry being the  $i^{\text{th}}$  bit of  $x_2$ .



**Fig. 1.** Relation  $\mathcal{R}_{\text{cds}}$  associated with CDS



**Fig. 2.** Relation  $\mathcal{R}_{\text{zk}}$  associated with ZKAoK

- Generate the second message of  $\text{GenProt}$ , i.e.,  $GP_2$ , as a function of  $GP_1$ , and freshly sampled randomness  $R_{gp}^{sen}$ . That is,  $GP_2 \leftarrow \text{Sen.MsgGen}[\text{GenProt}](K, GP_1; R_{gp}^{sen})$ .
- Compute  $s = \text{PRF}(K, 2) \oplus x_2$ .



- Generate the first message of CDS protocol, denoted by  $CDS_1$ , as a function of instance  $\mathbf{y} = (OT_1, s, GP_1, GP_2)$ , witness  $w = (x_2, R_{ot})$  and randomness  $R_{cds}^{rec}$ . That is,  $CDS_1 \leftarrow \text{Rec.MsgGen}(\mathbf{y}, w; R_{cds}^{rec})$ . It sends  $(ZK_2, OT_1, GP_2, CDS_1, s)$  to  $P_1$ .

3.  $P_1 \rightarrow P_2$ :  $P_1$  computes the final message as follows:
  - Execute  $\text{gcsk} \leftarrow \text{GC.Gen}(1^\lambda; R_{gc})$ , where  $R_{gc}$  is the randomness used in the algorithm. Execute  $\langle \mathbf{k} \rangle = (\mathbf{k}_1, \dots, \mathbf{k}_{\ell_2}) \leftarrow \text{GC.Grbl}(\text{gcsk})$ , where  $\ell_2$  is the input length of party  $P_2$ . For every  $i \in [\ell_2]$ , we have  $\mathbf{k}_i = (k_i^0, k_i^1)$ .
  - It computes the garbled circuit  $\widehat{C} \leftarrow \text{GrbC}(\text{gcsk}, C)$ , where  $C$  is a boolean circuit defined as  $C(y) = f(x_1, y)$ , where  $y$  is of length  $\ell_2$ .
  - It computes the second message of OT as a function of first message and randomness  $R_{ot}^{sen}$ . That is,  $OT_2 \leftarrow \text{Sen.MsgGen}[\text{OT}](\langle \mathbf{k} \rangle, OT_1; R_{ot}^{sen})$ .
  - It computes the second message of CDSProt as a function of first message  $CDS_1$ , instance  $\mathbf{y}$  (its computed the same way as  $P_2$  does), secret  $s = \widehat{C}$  and randomness  $R_{cds}^{sen}$ . That is,  $CDS_2 \leftarrow \text{Sen.MsgGen}[\text{CDSProt}](CDS_1, \mathbf{y}, s; R_{cds}^{sen})$ .
  - It computes the final message of ZK, namely  $ZK_3$ . This is computed as a function of instance  $(CDS_1, CDS_2, OT_1, OT_2)$  and witness  $(R_{gc}, R_{ot}^{sen}, R_{cds}^{sen}, \widehat{C})$ .

Finally,  $P_2$  recovers  $\text{out}$  from its view using the algorithm in Figure 3.

**Output Computation:**

$P_2$  computes the following:

- Checks if the verifier of ZK accepts on input the transcript of ZK. If the check fails, abort.
- From the transcript of OT protocol, recover the value  $\langle \mathbf{k} \rangle_{x_2} = (k_1^{b_1}, \dots, k_{\ell_2}^{b_{\ell_2}})$ .
- Recover the garbled circuit  $\widehat{C}$  from the transcript of the CDS protocol.
- Finally, execute the evaluation algorithm  $\text{EvalGC}(\widehat{C}, \langle \mathbf{k} \rangle_{x_2})$  to obtain the value  $\text{out}$ .

Output  $\text{out}$ .

**Fig. 3.** Computation of Output

**Theorem 12.** *Assuming the security of the following primitives: garbling scheme GC, oblivious transfer protocol OT, ZKAoK system ZK, generation protocol GenProt, conditional disclosure of secrets protocol CDSProt and pseudorandom functions PRF, we have that  $\Pi_{2PC}$  is secure against malicious adversaries (Definition 1).*

The proof of the above theorem can be found in the full version.

Instantiating the building blocks (see Section 3), we obtain the following corollary.

**Corollary 1.** *Assuming DDH, LWE, Zaps and succinct randomized encodings, protocol  $\Pi_{2PC}$  is a secure  $\mu$ -bounded uniform two party computation protocol satisfying Definition 2.*

## References

1. William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*, pages 119–135, 2001.
2. Prabhakaran Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 308–326, 2015.
3. Boaz Barak. How to go beyond the black-box simulation barrier. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 106–115. IEEE, 2001.
4. Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2001.
5. Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS '12*, pages 784–796, New York, NY, USA, 2012. ACM.
6. Mihir Bellare and Adriana Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, pages 273–289, 2004.
7. Nir Bitansky, Zvika Brakerski, Yael Tauman Kalai, Omer Paneth, and Vinod Vaikuntanathan. 3-message zero knowledge against human ignorance. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part I*, pages 57–83, 2016.
8. Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the existence of extractable one-way functions. *SIAM Journal on Computing*, 45(5):1910–1952, 2016.
9. Nir Bitansky, Sanjam Garg, Huijia Lin, Rafael Pass, and Siddhartha Telang. Succinct randomized encodings and their applications. In *STOC*, 2015.
10. Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *Advances in Cryptology-ASIACRYPT 2013*, pages 280–300. Springer, 2013.
11. Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In *Public-Key Cryptography-PKC 2014*, pages 501–519. Springer, 2014.

12. Ran Canetti, Justin Holmgren, Abhishek Jain, and Vinod Vaikuntanathan. Indistinguishability obfuscation of iterated circuits and RAM programs. In *STOC*, 2015.
13. Nico Döttling, Nils Fleischhacker, Johannes Krupp, and Dominique Schröder. Two-message, oblivious evaluation of cryptographic functionalities. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III*, pages 619–648, 2016.
14. Cynthia Dwork and Moni Naor. Zaps and their applications. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 283–293, 2000.
15. Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. In *Advances in Cryptology: Proceedings of CRYPTO '82, Santa Barbara, California, USA, August 23-25, 1982.*, pages 205–210, 1982.
16. Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 416–426, 1990.
17. Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 40–49. IEEE Computer Society, 2013.
18. Sanjam Garg, Pratyay Mukherjee, Omkant Pandey, and Antigoni Polychroniadou. The exact round complexity of secure computation. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, pages 448–476, 2016.
19. Yael Gertner, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. Protecting data privacy in private information retrieval schemes. *J. Comput. Syst. Sci.*, 60(3):592–629, 2000.
20. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM (JACM)*, 33(4):792–807, 1986.
21. Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for NP. *J. Cryptology*, 9(3):167–190, 1996.
22. Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM J. Comput.*, 25(1):169–192, 1996.
23. Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. In *Foundations of Computer Science, 1986., 27th Annual Symposium on*, pages 174–187. IEEE, 1986.
24. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game. In *STOC*, 1987.
25. Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *J. Cryptology*, 7(1):1–32, 1994.
26. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. In *STOC*, pages 291–304, 1985.
27. Satoshi Hada and Toshiaki Tanaka. On the existence of 3-round zero-knowledge protocols. In *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, pages 408–423, 1998.

28. Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 294–304. IEEE, 2000.
29. Yael Tauman Kalai, Guy N. Rothblum, and Ron D. Rothblum. From obfuscation to the security of fiat-shamir for proofs. In *CRYPTO*, 2017.
30. Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, pages 335–354, 2004.
31. Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 669–684. ACM, 2013.
32. Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for turing machines with unbounded memory. In *STOC*, 2015.
33. Dror Lapidot and Adi Shamir. Publicly verifiable non-interactive zero-knowledge proofs. In *Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings*, pages 353–365, 1990.
34. Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 52–78. Springer, 2007.
35. Moni Naor and Benny Pinkas. Oblivious transfer and polynomial evaluation. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 245–254. ACM, 1999.
36. Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, January 7-9, 2001, Washington, DC, USA.*, pages 448–457, 2001.
37. Michael O Rabin. How to exchange secrets with oblivious transfer. *IACR Cryptology ePrint Archive*, 2005:187, 2005.
38. Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 475–484. ACM, 2014.
39. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167, 1986.