On Secure Two-Party Computation in Three Rounds

Prabhanjan Ananth[1(✉)] and Abhishek Jain[2]

[1] University of California Los Angeles, Los Angeles, USA
prabhanjan@cs.ucla.edu
[2] Johns Hopkins University, Baltimore, USA
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 [24,39] 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

P. Ananth—Supported in part by grant 360584 from the Simons Foundation.
A. Jain—Supported in part by a DARPA/ARL Safeware Grant W911NF-15-C-0213 and a sub-award from NSF CNS-1414023.

Y. Kalai and L. Reyzin (Eds.): TCC 2017, Part I, LNCS 10677, pp. 612–644, 2017.
https://doi.org/10.1007/978-3-319-70500-2_21

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 [15,37]) with constant-round zero-knowledge proofs [16,21,26] 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.[1]

On Bounded Auxiliary Inputs. Our positive result is motivated by the recent beautiful works of [7,8] 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 Π 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 Π.*

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

[1] 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.[2]

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.

[2] 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].[3]

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

[3] 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.

as 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 $(\beta, \mathsf{trap}, m)$ from P_2. It treats TM as a valid Turing machine and computes $\beta' = \mathsf{TM}(m)$. If $\beta' = \beta$, it outputs trap, else it outputs $\perp$.[4] Let Π 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 Π computed using input $\mathsf{TM} = 0$. Let msg_1 denote this message. Upon receiving msg_1, P_2 first prepares an input tuple $(\beta, \mathsf{trap}, m)$ for g as follows: it samples a random string β of length ℓ s.t. $\ell \gg |\mathsf{msg}_1|$ and sets trap to be a random string and $m = \mathsf{msg}_1$. Finally, P_2 sends the second message of Π computed using $(\beta, \mathsf{trap}, m)$ together with β.

A non-black-box simulator that knows the Turing machine description TM_2 of adversarial P_2 can set its input $\mathsf{TM} = \mathsf{TM}_2$ in the above protocol. If P_2 behaves semi-honestly, then at the end of the protocol, the simulator should obtain trap. Security against a malicious P_1 can be argued using the fact that $\beta \gg |\mathsf{msg}_1|$ 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 Π 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 Π 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).[5] Let M be an "input-less" Turing machine that has hardwired in its description a tuple $(\mathsf{TM}, \beta, \mathsf{trap}, m)$. 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

[4] Note that g internally transforms TM into a circuit and uses it to perform the rest of the computation.

[5] 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 [1,19]. 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 created "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 Π for computing f, if Π achieves $2^{O(L)}$-indistinguishability security against P_1. Here, L is the length of the first message of Π.

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 Π 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 msg_1 from P_1 as input, the program honestly computes the protocol message msg_2 of P_2 (as per protocol Π) using input x_2 and randomness $r = F(K, \mathsf{msg})$, where F is another PRF.
2. Upon receiving a protocol transcript $(\mathsf{msg}_1, \mathsf{msg}_2, \mathsf{msg}_3)$, it re-computes the randomness r used to compute 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 msg_1 from P_1, it runs the obfuscated program on msg_1 to obtain msg_2 and then forwards it to P_1. Finally, upon receiving msg_3 from P_1, it submits the protocol transcript $(\mathsf{msg}_1, \mathsf{msg}_2, \mathsf{msg}_3)$ to the obfuscated program to obtain an output y.

Proof Strategy: Attempt #1. For any simulator S for Π, 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 Π 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 msg_1 of P_1. In the i^{th} hybrid, we use an auxiliary input Z_i that has both x_2 and x_2' hardwired inside it. On input first messages $\mathsf{msg}_1 < i$, 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 Π 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 [6,27] using "knowledge assumptions." More recently, [7,8] 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 λ. 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 $\mathcal{D}_1 \cong_{c,\varepsilon} \mathcal{D}_2$ if the following

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

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

If ε 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: $x \in L(R)$ 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 *μ-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 Π between parties P_1 and P_2. We define the notation $P_1.\mathsf{MsgGen}[\Pi]$ (resp., $P_2.\mathsf{MsgGen}[\Pi]$) to denote the algorithm that generates the next message of P_1 (resp., P_2). The notation $\beta \leftarrow P_1.\mathsf{MsgGen}[\Pi](\alpha, \mathsf{st}; r)$ indicates that the output of next message algorithm of party P_1 on input α, current state st and randomness r is the string β. Initially, st is set to $\perp$. For convenience of notation, we assume that the $\mathsf{MsgGen}[\cdot]$ is a stateful algorithm and hence, we avoid describing the parameter 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 $\mathsf{View}_{P,y}$.

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 Π_{2PC} secure against malicious adversaries.[6] 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[7]. 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 $\mathtt{out} = f(x_1', x_2')$ to P_2.
5. **Output:** If the party P_2 is honest, then it outputs $\mathtt{out}$. 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 $\mathsf{Ideal}_{f,\mathcal{B}}^{\Pi_{\mathsf{2PC}}}(x_1, x_2, \mathbf{z})$ to be the joint distribution over the outputs of the adversary and the honest party[8].

REAL PROCESS: In the real process, both the parties execute the protocol Π_{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 $\mathsf{Real}_{f,\overrightarrow{P}}^{\Pi_{\mathsf{2PC}}}(x_1, x_2, \mathbf{z})$ 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 Π_{2PC} be a two party protocol implementing f. We say that Π_{2PC}*

[6] 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.

[7] This means that at most one of the parties could deviate from the rules prescribed by the ideal process.

[8] If P_1 is honest, it does not have any output.

securely 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* $\mathbf{z} \in \{0,1\}^{\mathrm{poly}(\lambda)}$,

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

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

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

- *For every* μ*-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* $\mathbf{z} \in \{0,1\}^{\mu(\lambda)}$,

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

- *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* $\mathbf{z} \in \{0,1\}^{\mathrm{poly}(\lambda)}$,

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

3 Building Blocks

We describe the building blocks used in our results.

3.1 Garbling Schemes

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

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

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

It satisfies the following properties:

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

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

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

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

where:
- $\mathsf{gcsk} \leftarrow \mathsf{Gen}(1^\lambda)$
- $\widehat{C} \leftarrow \mathsf{GrbC}(\mathsf{gcsk}, C)$
- $((k_1^0, k_1^1), \ldots, (k_\ell^0, k_\ell^1)) \leftarrow \mathsf{GrbI}(\mathsf{gcsk})$
- $\phi(C)$ *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 [15,37] 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 $b_{\overline{c}}$ 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 $b' \leftarrow \langle \mathsf{Sen}(b_0, b_1),\ \mathsf{Rec}(c)\rangle$.*

We require that an OT protocol satisfies the following properties:

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

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

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

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

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

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

We define ℓ*-parallel 1-out-2 OT to be a protocol that is composed of* ℓ *parallel executions of 1-ou-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* OT *between two parties* P_1 *(sender) and* P_2 *(receiver). We say that* 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* $\mathsf{Transcript} = (OT_1, \ldots, OT_k)$*. 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* $\mathsf{Transcript}$ *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 OT_n, is composed of n parallel copies of OT and if OT satisfies uniqueness of transcript property then so does OT_n. In particular, given the randomness of the receiver of 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 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 $c_{\mathsf{bit}} = ab$. It generates $c_{1-\mathsf{bit}}$ at random such that $c_{\mathsf{bit}} \neq c_{1-\mathsf{bit}}$. 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* Ext *(not necessarily efficient) such the following holds. Consider the following experiment:* $\underline{\mathsf{Expt}(1^\lambda, b)}$*:*

- *P_1^* outputs the first message* msg_1.
- *Extractor* Ext *on input* 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* msg_1 *and x_2^b, outputs the second message* msg_2.
- *P_1^* upon receiving the second message outputs a bit* $\mathtt{out}$.
- *Output* $\mathtt{out}$.

We require that,

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

for some negligible function 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_2^1*. 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 ℓ_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 ℓ_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 $\mathsf{Gen}(1^\lambda)$ to get gcsk.
  - It then computes $\mathsf{GrbC}(\mathsf{gcsk}, C)$ to obtain $\widehat{C}$. C is a circuit with x_2 hardwired in it; it takes as input x_1 and computes $f(x_1, x_2)$.
  - It computes $\mathsf{GrbI}(\mathsf{gcsk})$ to obtain the wire keys $(\mathbf{k}_1, \ldots, \mathbf{k}_{\ell_1})$, 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 $(\mathbf{k}_1, \ldots, \mathbf{k}_{\ell_1})$.

  It sends $(\widehat{C}, OT_2)$ to P_1.
- P_1: Upon receiving $(\widehat{C}, OT_2)$, it recovers the wire keys $(k_1, \ldots, k_{\ell_1})$. It then executes $\mathsf{EvalGC}(\widehat{C}, (k_1, \ldots, k_{\ell_1}))$ 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.

$\underline{\mathsf{Hyb}_{1.b}}$ for $b \xleftarrow{\$} \{0,1\}$: 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.

$\underline{\mathsf{Hyb}_{2.b}}$ for $b \xleftarrow{\$} \{0,1\}$: Let x_1^* be the input extracted by the extractor. We denote the i^{th} bit of x_1^* to be $x_{1,i}^*$. As part of the second message, the wire keys $(\mathbf{k}_1, \ldots, \mathbf{k}_{\ell_1})$, 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 $(\mathbf{k}_1, \ldots, \mathbf{k}_{\ell_1})$, it generates OT_2 as a function of $(\mathbf{k}'_1, \ldots, \mathbf{k}'_{\ell_1})$. $\mathbf{k}'_i$ contains $\left(0, k_i^{x_{1,i}^*}\right)$ if $x_{1,i}^* = 1$, otherwise it contains $\left(k_i^{x_{1,i}^*}, 0\right)$.

Hybrids $\mathsf{Hyb}_{1.b}$ and $\mathsf{Hyb}_{2.b}$ are computationally distinguishable from the indistinguishability security against malicious receivers property of the oblivious transfer protocol.

$\underline{\mathsf{Hyb}_{3.0}}$: 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 $\mathsf{Hyb}_{2.b}$ and $\mathsf{Hyb}_{3.0}$ are computationally indistinguishable from the security of the garbling schemes[9].

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) [1,19] 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.

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* $s \leftarrow \langle P_1(\mathbf{X}, w),\ P_2(\mathbf{X}, s)\rangle$*.*

[9] Formally this is argued by first simulating the garbled circuit and then switching the input.

– **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,*

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

for some negligible function 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 (Sect. 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 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* $(\mathsf{Prover}, \mathsf{Verifier})$ *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* $\langle \mathsf{Prover}(\mathbf{y}, \mathbf{w}),\ \mathsf{Verifier}(\mathbf{y})\rangle$*. It consists of the following properties:*

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

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

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

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

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

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

We define a ZKAoK system to be k*-message if the number of messages between* Prover *and* $\mathsf{Verifier}$ *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 (μ-Bounded Uniform Zero Knowledge). *A proof system* $(\mathsf{Prover}, \mathsf{Verifier})$ *for a relation* $\mathcal{R}$ *is said to be* μ**-bounded uniform ZKAoK** *if the following holds:*

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

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

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* μ*-bounded uniform zero knowledge for some function* μ*, 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* $\mathsf{SRE} = (\mathsf{E}, \mathsf{D})$ *for a class of Turing machines* $\mathcal{M}$ *consists of the following probabilistic polynomial time algorithms:*

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

We require that the above algorithms satisfies the following properties:

- **Correctness***: We require that the following holds for every* $M \in \mathcal{M}, x \in \{0,1\}^*$*,*
$$\mathsf{Pr}\left[\mathsf{D}(\langle M, x \rangle) = M(x) \ : \ \langle M, x \rangle \leftarrow \mathsf{E}(1^\lambda, M, x)\right] = 1$$
- **Security***: For every PPT adversary* $\mathcal{A}$*, there exists a PPT simulator* Sim *such that the following holds:*
$$\{\langle M, x \rangle\} \approx_c \left\{\mathsf{Sim}(1^\lambda, 1^{|M|}, 1^{|x|}, M(x))\right\},$$
where:
  - $\langle M, x \rangle \leftarrow \mathsf{E}(1^\lambda, M, x)$

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 $\mathtt{out}$. We denote the randomized encoding of an input-less TM to be $\langle M \rangle \leftarrow \mathsf{E}(1^\lambda, M, \perp)$.

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* iO *is called an* ε*-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*
$$\mathsf{Pr}\left[C'(x) = C(x) \ : \ C' \leftarrow \mathsf{iO}(\lambda, C)\right] = 1$$

– **ε-Indistinguishability:** *For any PPT distinguisher D, there exists a negligible function* $\mathsf{negl}(\cdot)$ *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:*

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

If ε is negligible in λ then we refer to 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 $\mathsf{PRF}_{punc}(K, \cdot)$ is said to be a μ-secure puncturable PRF if there exists a PPT algorithm* $\mathsf{Puncture}$ *that satisfies the following properties:*

– ***Functionality preserved under puncturing.*** $\mathsf{Puncture}$ *takes as input a PRF key K and an input x and outputs $K\backslash\{x\}$ such that for all $x' \neq x$, $\mathsf{PRF}_{punc}(K\backslash\{x\}, x') = \mathsf{PRF}_{punc}(K, x')$.*
– **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 $K\backslash\{x\} \leftarrow \mathsf{Puncture}(K, x)$. Then for all sufficiently large $\lambda \in \mathbb{N}$,*

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

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

If μ is negligible, we refer to PRF_{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 ([10,11,20,31]). *If $\frac{\mu}{\mathrm{poly}}$-secure one-way functions exist, for some fixed polynomial* poly*, then there exists μ-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[10] 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 $K \in \{0,1\}^{\mathrm{poly}(\lambda)}$. 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 $K_0, K_1 \in \{0,1\}^{\mathrm{poly}(\lambda)}$ and auxiliary information $\mathbf{z} \in \{0,1\}^{\mathrm{poly}'(\lambda)}$,*

$$\left|\mathsf{Pr}\left[1 \leftarrow \langle P_1^*(\mathbf{z}, K_0, K_1),\ P_2(\mathbf{z}, K_0)\rangle\right] - \mathsf{Pr}\left[1 \leftarrow \langle P_1^*(\mathbf{z}, K_0, K_1),\ P_2(\mathbf{z}, K_1)\rangle\right]\right| \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

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

[10] 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 Against μ-Bounded Uniform Senders. We consider generation protocols where the extractability property needs to hold against senders modeled as μ-bounded uniform algorithms. We formally define this below.

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

- *It satisfies the soundness property in Definition 16.*
- **Extractability against μ-bounded uniform senders***: For every semi-honest PPT P_2 (modeled as a Turing machine), there exists a PPT extractor* 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* $\mathbf{z} \in \{0,1\}^{\mu(\lambda)}$*,*

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

Remark 4. If μ 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 μ-bounded uniform senders.

Tools. The first tool we use is succinct randomized encodings for Turing machines, denoted by $\mathsf{SRE} = (\mathsf{E}, \mathsf{D})$. Another tool we use is a two message secure function evaluation protocol Π_{2PC}. In particular, we use the two message secure function evaluation protocol defined in Sect. 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 Π_{2PC}: The functionality f associated with Π_{2PC} is the following: f on input $x_2 = (\beta, K, m, R_2, \mathsf{md}, \theta)$ 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 $\mathsf{md} = 1$ then compute the succinct randomized encoding $\langle N\rangle \leftarrow \mathsf{E}(1^\lambda, N[\beta, K, m, M], \bot; R)$ (i.e., R is the randomness used in E), where R is set to $R_1 \oplus R_2$. The Turing machine N is an input-less Turing machine (refer Sect. 3.6) that does the following: hardwired inside it are the values (β, K, m, M).
  1. It first computes $M(m)$ to get as output $\mathtt{out}$.
  2. It interprets the first $|\beta|$ number of bits of $\mathtt{out}$ to be the string β'.
  3. It checks if $\beta' = \beta$. If so, it outputs K. Otherwise, it outputs $\bot$.

  It outputs $\langle N\rangle$.
- If $\mathsf{md} = 2$ then:
  1. It outputs θ.

Construction. We describe the protocol below. Denote the receiver to be P_1 and the sender to be P_2. Call this protocol $\mathsf{GenProt}$.

- Upon input $\mathbf{z}$, P_1 (receiver) prepares an input x_1 for Π_{2PC} as a $\mu(\lambda)$-length string of all zeroes. It takes the role of the party $\mathcal{P}_1$ in the protocol Π_{2PC}. It computes the first message $\mathbf{msg}_1$ of Π_{2PC} using the input x_1. That is, $\mathbf{msg}_1 \leftarrow \mathcal{P}_1.\mathsf{MsgGen}[\Pi_{\mathsf{2PC}}](1^\lambda, x_1)$. It sends $\mathbf{msg}_1$ to P_2 (sender).
- Upon input $\mathbf{z}$ and trapdoor K, P_2 (sender) first picks a string β of length $\ell_\beta = \mathrm{poly}(\lambda)$ 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 Π_{2PC}. It then sets its input to Π_{2PC} to be $x_2 = (\beta, K, \mathbf{msg}_1, R, \mathsf{md}, \theta)$, where $\mathsf{md} = 1$ and $\theta = 0$. Using x_2 and $\mathbf{msg}_1$, it computes the second message $\mathbf{msg}_2$ of Π_{2PC} using the input x_2. That is, $\mathbf{msg}_2 \leftarrow \mathcal{P}_2.\mathsf{MsgGen}[\Pi_{\mathsf{2PC}}](1^\lambda, x_2, \mathbf{msg}_1)$. It sends $(\beta, \mathbf{msg}_2)$ to P_1.

Finally, P_1 computes the output of Π_{2PC} and recovers the randomized encoding $\langle N \rangle$. It then evaluates the decoding algorithm $\mathsf{D}(\langle N \rangle)$ 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 Π_{2PC} (Definition 7) and* SRE, $\mathsf{GenProt}$ *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 Sim, for every $K \in \{0,1\}^{\mathrm{poly}(\lambda)}$ and auxiliary information $\mathbf{z} \in \{0,1\}^{\mathrm{poly}'(\lambda)}$,

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

Note that the above property implies soundness property.

Description of $\mathsf{Sim}(\mathbf{z})$. It receives as input msg_1 from P_1. It generates msg_2 as follows:

- Let $\mathsf{Sim}_{\mathsf{SRE}}$ be the simulator of the succinct randomized encodings scheme. It then executes $\mathsf{Sim}_{\mathsf{SRE}}(1^\lambda, 1^{\ell_1}, 1^{\ell_2}, v)$, where ℓ_1 is the size of M, ℓ_2 is the size of m as defined in the description of functionality for Π_{2PC} and v is set to be $\perp$. The output of $\mathsf{Sim}_{\mathsf{SRE}}(1^\lambda, 1^{\ell_1}, 1^{\ell_2}, v)$ is denoted by $\langle N \rangle$.
- It sets $x_2 = (0, 0, 0, 0, 2, \langle N \rangle)$. It then computes msg_2 as a function of x_2 and msg_1. The generation of msg_2 is performed by running the algorithm of (honest) $\mathcal{P}_2$ in Π_{2PC}. That is, $\mathsf{msg}_2 \leftarrow \mathcal{P}_2.\mathsf{MsgGen}[\Pi_{\mathsf{2PC}}](1^\lambda, x_2, \mathsf{msg}_1)$.
- Finally, it samples a string β of length ℓ_β.

Sim then sends (β, 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 Π_{2PC} (see Definition 6). Recall that Ext need not necessarily be efficient.

Hyb_1: This corresponds to the real experiment where $P_1^*(\mathbf{z})$ is interacting with $P_2(\mathbf{z}, K)$. The output of this hybrid is the output of P_1^*.

Hyb_2: 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 Π_{2PC}. That is, $\mathsf{msg}_2 \leftarrow \mathcal{P}_2.\mathsf{MsgGen}[\Pi_{\mathsf{2PC}}](1^\lambda, x_2', \mathsf{msg}_1)$, where msg_1 is the message sent by P_1^*. Here, θ is set to be the output $f(x_1^*, (\beta, K, \mathsf{msg}_1, R_2, 1, 0))$, where R_2 is sampled uniformly at random. P_2 sends (β, msg_2) to P_1^*, where β is a string of length ℓ_β 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 Π_{2PC}, hybrids Hyb_1 and Hyb_2 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_1 and let x_2'' be the input used by $\mathcal{P}_2$ in Hyb_2. We have that $f(x_1, x_2') = f(x_1, x_2'')$. And thus, from the security of Π_{2PC} (Definition 6), we have that P_1^* cannot distinguish whether P_2 used x_2' or x_2''. The claim thus follows.

Hyb_3: In this hybrid, P_2 essentially executes the simulator Sim described above.

Claim. Assuming the security of SRE, hybrids Hyb_2 and Hyb_3 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 Ext from the first message msg_1 of Π_{2PC}. Sample string β of length ℓ_β uniformly at random. We first make the following observation. The probability that for any γ, $M(\gamma)$ outputs the random string β is at most $2^{-\mathcal{O}(\ell_\beta - \mu(\lambda) - \lambda)}$, which is negligible. Thus with overwhelming probability we have that $N[\beta, K, \mathsf{msg}_1, M]$ outputs $\perp$.

The only difference between Hyb_2 and Hyb_3 is that in Hyb_2, θ is set to $\langle N \rangle$ whereas in Hyb_3, θ 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 Hyb_2 and Hyb_3 are computationally indistinguishable.

From the indistinguishability of Hyb_1 and Hyb_3, we have that P_1^* cannot distinguish whether it is interacting with P_2 versus interacting with Sim. This completes the proof.

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

Proof. We design an extractor 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 μ-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 λ.

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

- From the security of Π_{2PC} (Definition 6), the view of P_2^* when interacting with P_1 is computationally indistinguishable from the view of P_2 when interacting with ExtGP. Recall that P_1 uses the input 0 in the first message and ExtGP uses the input M in the first message.
- Since P_2^* is semi-honest, it computes the second message of Π_{2PC} honestly. From the correctness of Π_{2PC}, it follows that the extractor can recover the randomized encoding $\langle N \rangle$ from Π_{2PC}. From the correctness of SRE, it further follows that the decoding of $\langle N \rangle$ yields K if and only if the first ℓ_β bits of $M(\mathbf{msg}_1)$ yields β. 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 $\mathsf{GenProt}$ 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 Π_{2PC} 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 $\mathsf{GC} = (\mathsf{Gen}, \mathsf{GrbC}, \mathsf{GrbI}, \mathsf{EvalGC})$. Without loss of generality we can assume that GrbC and GrbI are deterministic algorithms.

2. Two message ℓ_2-parallel 1-out-2 oblivious transfer protocol (Definition 4), denoted by OT. We require security against malicious receivers.

We additionally require that the 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 $\mathsf{ZK} = (\mathsf{Prover}, \mathsf{Verifier})$ satisfies the delayed statement-witness property (Definition 11).

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

4. Two Message Generation Protocol (Definition 16) denoted by $\mathsf{GenProt}$. 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 $\mathsf{GenProt}$ is played by P_2 and the role of the receiver of $\mathsf{GenProt}$ is played by P_1.

5. Two Message Conditional Disclose of Secret (CDS) Protocol (Definition 8), denoted by $\mathsf{CDSProt}$. The associated relation $\mathcal{R}_{\mathsf{cds}}$ is described in Fig. 1.

6. Other tools. We additionally use pseudorandom functions, denoted by PRF, in this construction.

Relation $\mathcal{R}_{\mathsf{cds}}$ for CDS Protocol

Input: $\mathbf{y} = (OT_1, s, GP_1, GP_2)$
Witness: $w = (x_2, R_{ot}^{rec}, K, R_{gp}^{sen})$

$(\mathbf{y}, w)$ is in relation $\mathcal{R}_{\mathsf{cds}}$ if and only if the following conditions are satisfied:

1. OT_1 is generated as a function of x_2 and R_{ot}^{rec}. That is, $OT_1 \leftarrow \mathsf{Rec.MsgGen}[\mathsf{OT}](x_2; R_{ot}^{rec})$.
2. The trapdoor K was used honestly to generate the message GP_1 using $\mathsf{GenProt}$. The randomness used by the sender in this protocol is R_{gp}^{sen}. That is, $GP_1 \leftarrow \mathsf{Sen.MsgGen}[\mathsf{GenProt}](K; R_{gp}^{sen})$.
3. $R_{ot}^{rec} \leftarrow \mathsf{PRF}(K, 1)$.
4. $s \leftarrow \mathsf{PRF}(K, 2) \oplus x_2$.

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

Relation $\mathcal{R}_{\mathsf{zk}}$ for ZKAoK Protocol

Input : $\mathbf{y} = (CDS_1, CDS_2, OT_1, OT_2)$
Witness : $w = (R_{gc}, R_{ot}^{sen}, R_{cds}^{sen}, \widehat{C})$

$(\mathbf{y}, w)$ is in relation $\mathcal{R}_{\mathsf{zk}}$ if and only if the following conditions are satisfied:

1. Garbling key is generated as follows; $\mathsf{gcsk} \leftarrow \mathsf{Gen}(1^\lambda; R_{gc})$. Garbled circuit $\widehat{C}$ is generated as $\widehat{C} \leftarrow \mathsf{GrbC}(\mathsf{gcsk}, C)$. Wire keys are generated as $\langle \mathbf{k} \rangle \leftarrow \mathsf{GrbI}(\mathsf{gcsk})$.
2. The second message of OT is generated as $OT_2 \leftarrow \mathsf{Sen.MsgGen}[\mathsf{OT}](\langle \mathbf{k} \rangle, OT_1; R_{ot}^{sen})$.
3. CDS_2 is computed as a function of CDS_1 and randomness R_{cds}^{sen}. That is, $CDS_2 \leftarrow \mathsf{Sen.MsgGen}[\mathsf{CDSProt}](CDS_1; R_{cds}^{sen})$.

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

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 $\langle \mathbf{k} \rangle_{x_2} = (k_1^{b_1}, \ldots, k_{\ell_2}^{b_{\ell_2}})$.
- Recover the garbled circuit $\widehat{C}$ from the transcript of the CDS protocol.
- Finally, execute the evaluation algorithm $\mathsf{EvalGC}(\widehat{C}, \langle \mathbf{k} \rangle_{x_2})$ to obtain the value out.

Output out.

Fig. 3. Computation of output

Protocol Π_{2PC}. We now proceed to describe protocol Π_{2PC}.

1. $P_1 \rightarrow P_2$: On input x_1 of length ℓ_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}^{rec}. That is, $GP_1 \leftarrow \mathsf{Rec.MsgGen}[\mathsf{GenProt}](R_{gp}^{rec})$.

   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 $R_{ot}^{rec} = \mathsf{PRF}(K, 1)$, randomness used in OT.
   - It computes the first message of OT, denoted by OT_1, as a function of its input x_2 and randomness R_{ot}^{rec}. That is, $OT_1 \leftarrow \mathsf{Rec.MsgGen}[\mathsf{OT}](x_2; R_{ot}^{rec})$, where Rec is the receiver algorithm of OT.

Here, x_2 is interpreted as a vector with the i^{th} entry being the i^{th} bit of x_2.
- Generate the second message of $\mathsf{GenProt}$, i.e., GP_2, as a function of GP_1, and freshly sampled randomness R_{gp}^{sen}. That is, $GP_2 \leftarrow \mathsf{Sen.MsgGen}[\mathsf{GenProt}](K, GP_1; R_{gp}^{sen})$.
- Compute $s = \mathsf{PRF}(K, 2) \oplus x_2$.
- 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, $CDS_1 \leftarrow \mathsf{Rec.MsgGen}(\mathbf{y}, w; R_{cds}^{rec})$.

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 $\mathsf{gcsk} \leftarrow \mathsf{GC.Gen}(1^\lambda; R_{gc})$, where R_{gc} is the randomness used in the algorithm. Execute $\langle \mathbf{k} \rangle = (\mathbf{k}_1, \ldots, \mathbf{k}_{\ell_2}) \leftarrow \mathsf{GC.Grbl}(\mathsf{gcsk})$, where ℓ_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 $\widehat{C} \leftarrow \mathsf{GrbC}(\mathsf{gcsk}, C)$, where C is a boolean circuit defined as $C(y) = f(x_1, y)$, where y is of length ℓ_2.
   - It computes the second message of OT as a function of first message and randomness R_{ot}^{sen}. That is, $OT_2 \leftarrow \mathsf{Sen.MsgGen}[\mathsf{OT}](\langle \mathbf{k} \rangle, OT_1; R_{ot}^{sen})$.
   - It computes the second message of $\mathsf{CDSProt}$ 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, $CDS_2 \leftarrow \mathsf{Sen.MsgGen}[\mathsf{CDSProt}](CDS_1, \mathbf{y}, s; R_{cds}^{sen})$.
   - 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 $\mathfrak{out}$ from its view using the algorithm in Fig. 3.

Theorem 12. *Assuming the security of the following primitives: garbling scheme* GC*, oblivious transfer protocol* OT*, ZKAoK system* ZK*, generation protocol* $\mathsf{GenProt}$*, conditional disclosure of secrets protocol* $\mathsf{CDSProt}$ *and pseudorandom functions* PRF*, we have that* Π_{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 Sect. 3), we obtain the following corollary.

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

References

1. Aiello, B., Ishai, Y., Reingold, O.: Priced oblivious transfer: how to sell digital goods. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 119–135. Springer, Heidelberg (2001). doi:10.1007/3-540-44987-6_8

2. Ananth, P., Jain, A.: Indistinguishability obfuscation from compact functional encryption. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 308–326. Springer, Heidelberg (2015). doi:10.1007/978-3-662-47989-6_15
3. Barak, B.: How to go beyond the black-box simulation barrier. In: Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science, pp. 106–115. IEEE (2001)
4. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S., Yang, K.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001). doi:10.1007/3-540-44647-8_1
5. Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS 2012, New York, NY, USA, pp. 784–796. ACM (2012)
6. Bellare, M., Palacio, A.: The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 273–289. Springer, Heidelberg (2004). doi:10.1007/978-3-540-28628-8_17
7. Bitansky, N., Brakerski, Z., Kalai, Y., Paneth, O., Vaikuntanathan, V.: 3-message zero knowledge against human ignorance. In: Hirt, M., Smith, A. (eds.) TCC 2016. LNCS, vol. 9985, pp. 57–83. Springer, Heidelberg (2016). doi:10.1007/978-3-662-53641-4_3
8. Bitansky, N., Canetti, R., Paneth, O., Rosen, A.: On the existence of extractable one-way functions. SIAM J. Comput. **45**(5), 1910–1952 (2016)
9. Bitansky, N., Garg, S., Lin, H., Pass, R., Telang, S.: Succinct randomized encodings and their applications. In: STOC (2015)
10. Boneh, D., Waters, B.: Constrained pseudorandom functions and their applications. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013. LNCS, vol. 8270, pp. 280–300. Springer, Heidelberg (2013). doi:10.1007/978-3-642-42045-0_15
11. Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 501–519. Springer, Heidelberg (2014). doi:10.1007/978-3-642-54631-0_29
12. Canetti, R., Holmgren, J., Jain, A., Vaikuntanathan, V.: Indistinguishability obfuscation of iterated circuits and RAM programs. In: STOC (2015)
13. Döttling, N., Fleischhacker, N., Krupp, J., Schröder, D.: Two-message, oblivious evaluation of cryptographic functionalities. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9816, pp. 619–648. Springer, Heidelberg (2016). doi:10.1007/978-3-662-53015-3_22
14. Dwork, C., Naor, M.: Zaps and their applications. In: 41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12–14 November 2000, Redondo Beach, California, USA, pp. 283–293 (2000)
15. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) CRYPTO 1982, pp. 205–210. Springer, Boston (1982). doi:10.1007/978-1-4757-0602-4_19
16. Feige, U., Shamir, A.: Witness indistinguishable and witness hiding protocols. In: Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, 13–17 May 1990, Baltimore, Maryland, USA, pp. 416–426 (1990)
17. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: 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, pp. 40–49. IEEE Computer Society (2013)

18. Garg, S., Mukherjee, P., Pandey, O., Polychroniadou, A.: The exact round complexity of secure computation. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 448–476. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49896-5_16
19. Gertner, Y., Ishai, Y., Kushilevitz, E., Malkin, T.: Protecting data privacy in private information retrieval schemes. J. Comput. Syst. Sci. **60**(3), 592–629 (2000)
20. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. ACM (JACM) **33**(4), 792–807 (1986)
21. Goldreich, O., Kahan, A.: How to construct constant-round zero-knowledge proof systems for NP. J. Cryptol. **9**(3), 167–190 (1996)
22. Goldreich, O., Krawczyk, H.: On the composition of zero-knowledge proof systems. SIAM J. Comput. **25**(1), 169–192 (1996)
23. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. In: 27th Annual Symposium on Foundations of Computer Science, pp. 174–187. IEEE (1986)
24. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: STOC (1987)
25. Goldreich, O., Oren, Y.: Definitions and properties of zero-knowledge proof systems. J. Cryptol. **7**(1), 1–32 (1994)
26. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems. In: STOC, pp. 291–304 (1985)
27. Hada, S., Tanaka, T.: On the existence of 3-round zero-knowledge protocols. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 408–423. Springer, Heidelberg (1998). doi:10.1007/BFb0055744
28. Ishai, Y., Kushilevitz, E.: Randomizing polynomials: a new representation with applications to round-efficient secure computation. In: Proceedings of the 41st Annual Symposium on Foundations of Computer Science, pp. 294–304. IEEE (2000)
29. Kalai, Y.T., Rothblum, G.N., Rothblum, R.D.: From obfuscation to the security of Fiat-Shamir for proofs. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10402, pp. 224–251. Springer, Cham (2017). doi:10.1007/978-3-319-63715-0_8
30. Katz, J., Ostrovsky, R.: Round-optimal secure two-party computation. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 335–354. Springer, Heidelberg (2004). doi:10.1007/978-3-540-28628-8_21
31. Kiayias, A., Papadopoulos, S., Triandopoulos, N., Zacharias, T.: Delegatable pseudorandom functions and applications. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, pp. 669–684. ACM (2013)
32. Koppula, V., Lewko, A.B., Waters, B.: Indistinguishability obfuscation for turing machines with unbounded memory. In: STOC (2015)
33. Lapidot, D., Shamir, A.: Publicly verifiable non-interactive zero-knowledge proofs. In: Menezes, A.J., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 353–365. Springer, Heidelberg (1991). doi:10.1007/3-540-38424-3_26
34. Lindell, Y., Pinkas, B.: An efficient protocol for secure two-party computation in the presence of malicious adversaries. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 52–78. Springer, Heidelberg (2007). doi:10.1007/978-3-540-72540-4_4
35. Naor, M., Pinkas, B.: Oblivious transfer and polynomial evaluation. In: Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, pp. 245–254. ACM (1999)

36. Naor, M., Pinkas, B.: Efficient oblivious transfer protocols. In: Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, 7–9 January 2001, Washington, D.C., USA., pp. 448–457 (2001)
37. Rabin, M.O.: How to exchange secrets with oblivious transfer. IACR Cryptology ePrint Archive, 2005:187 (2005)
38. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: Shmoys, D.B. (ed.) Symposium on Theory of Computing, STOC 2014, New York, NY, USA, 31 May–03 June 2014, pp. 475–484. ACM (2014)
39. Yao, A.C.-C.: How to generate and exchange secrets (extended abstract). In: FOCS, pp. 162–167 (1986)