What Security Can We Achieve In 4-Rounds?

Carmit Hazay[*] Muthuramakrishnan Venkitasubramaniam[†]

Abstract

In this paper we study the question of what security is achievable for stand-alone two-party computation in four-rounds. Our starting point point is the Katz-Ostrovsky lower bound [KO04] which determines that the exact round complexity of achieving a secure two-party computation protocol is five. To get around this lower bound we consider two relaxations of the standard simulation-based security definition, where each relaxation implies a different security guarantee.

Specifically, we analyze our protocols in the presence of malicious non-aborting adversaries (for which we obtain full security) and malicious aborting adversaries (for which we obtain $1/p$-security, which implies that the simulation fails with probability at most $1/p + \mathsf{negl}$). We further prove that our security guarantee is tight with respect to the party that obtains the input first.

Keywords: Secure Computation, Coin-Tossing, Oblivious Transfer, Round Complexity

[*]Faculty of Engineering, Bar-Ilan University, Israel. Email: `carmit.hazay@biu.ac.il`.

[†]University of Rochester, Rochester, NY 14611, NY. Email: `muthuv@cs.rochester.edu`.

1 Introduction

Secure two-party computation enables two parties to mutually run a protocol that computes some function f on their private inputs, while preserving a number of security properties. Two of the most important properties are privacy and correctness. The former implies data confidentiality, namely, nothing leaks by the protocol execution but the computed output. The latter requirement implies that the protocol enforces the integrity of the computations made by the parties, namely, honest parties learn the correct output. Feasibility results are well established [Yao86, GMW87, MR91, Bea91], proving that any efficient functionality can be securely computed under full simulation-based definitions (following the ideal/real paradigm). Security is typically proven with respect to two adversarial models: the semi-honest model (where the adversary follows the instructions of the protocol but tries to learn more than it should from the protocol transcript), and the malicious model (where the adversary follows an arbitrary polynomial-time strategy), and feasibility holds in the presence of both types of attacks.

An important complexity measure of secure computation that has been extensively studied in literature, is the *round-complexity* of secure protocols. In the *stand-alone* setting, Yao [Yao86] presented the first constant-round secure two-party computation protocol in the semi-honest model. In contrast, Goldreich, Micali and Wigderson [GMW87] showed how to obtain protocols that tolerate malicious adversaries which requires non-constant number of rounds, followed by Lindell [Lin01] who gave the first constant-round secure two-party protocol tolerating such attacks. In an important characterization, Katz and Ostrovsky [KO04] determined that the exact round complexity of achieving a secure two-party computation protocol is five (and four if only one of the parties receives an output). More precisely, they constructed a five-round protocol to securely compute arbitrary functionalities and showed that there cannot exist any four-round black-box construction that securely realizes the coin-tossing functionality. More recently, Ostrovsky, Richelson and Scafuro [ORS15] strengthened this construction by demonstrating a five-round protocol where the underlying cryptographic primitives are used only in a "black-box" way. Both the results also provide a four-round protocol for single-output functionalities. While these results only consider the stand-alone model, assuming some trusted-setup such as a common reference string (CRS), it is possible to construct round-optimal (i.e. two-round) secure two-party protocols; see [HK07] for just one example.

In the context of secure two-party computation, zero-knowledge is a fundamental example of a two-party functionality [GMR89] for which its round complexity has been widely studied, starting with the work of Goldreich and Oren [GO94] who showed that two-round computational zero-knowledge proofs are impossible for languages outside BPP. Goldreich and Krawczyk [GK96] extended this impossibility result to three-round protocols that are black-box zero-knowledge, whereas Katz [Kat12] showed that only languages in MA admit four-round zero-knowledge proofs. On the positive side, Feige and Shamir [FS90] demonstrated how to achieve four-round zero-knowledge *arguments* for any language in NP.[1] A relaxation of zero-knowledge, referred to as *witness-indistinguishable* proofs only requires that no malicious verifier be able to distinguish which witness is used in the proof. For this relaxed notion, Feige and Shamir [FS90] showed how to construct three-round protocols. Dwork and Naor [DN07] showed that starting from a non-interactive zero-knowledge proof it is possible to construct a two-round witness-indistinguishable proofs (or ZAPs). Moreover, based on non-standard assumptions, Barak, Ong and Vadhan [BOV07] showed how to construct non-interactive witness-indistinguishable proofs.

Motivated by the progress in zero-knowledge proofs, the main question we address in this work is whether it is feasible to achieve any meaningful security for two-party computation in four-rounds. Specifically, in this work we investigate the following question:

What security is achievable for stand-alone two-party computation in four-rounds?

[1]Loosely speaking, an argument is an interactive proof system where the soundness property is only required to hold against efficient adversaries.

Towards understanding this question, we begin with the observation that the security model considered in [KO04] for proving their lower bound, rules out black-box four-round secure protocols for the coin-tossing functionality. An important artifact of the lower bound is that they only consider simulators that are allowed to make a fixed polynomial number of oracle queries to the adversary, which, in particular, cannot depend on the adversary's behavior (e.g., running time or abort probability). To get around their lower bound, it is quite conceivable that it is possible to construct a four-round protocol, as well as a simulator whose running time is polynomially related to, say for instance, the inverse of the abort probability of the adversary. However, such a scenario might require a priori "non-black-box" information regarding the adversary and we do not pursue this approach in this work. Instead, we consider two relaxations of the standard simulation-based security definition that get around this lower bound, where each relaxation implies a different security guarantee.

Concretely, the first relaxation is to only consider non-aborting adversaries. We stress that Katz-Ostrovsky crucially relies in their lower bound on aborting adversaries as they construct an adversary which aborts with probability that is correlated with the number of queries made by the simulation. Hence, ruling out such attack strategies invalidates their lower bound. In particular, we note that when considering non-aborting adversaries, security is still required in the presence of malicious adversaries with simulation-based security.

A second relaxation we consider is to weaken the indistinguishability requirement. Namely, in the real/ideal paradigm definition when comparing an ideal simulated execution to the real execution, this relaxation implies that the ideal execution is defined as in the original definition yet the simulation notion is relaxed. More concretely, the two executions are now required to be distinguishable with probability at most $\frac{1}{p} + \mathsf{negl}$, where $p(\cdot)$ is some specified polynomial. This relaxation has been considered in the past in the context of achieving coin-tossing [Cle86, MNS09] and fairness for arbitrary functionalities [GK10]. Then, in case of malicious (possibly aborting) adversaries we require that our protocol admits $\frac{1}{p}$-security.

To summarize this discussion, our protocols enjoy the best of both worlds given the limitations imposed by [KO04]. Namely, they achieve full simulation in the presence of non-aborting attack strategies, and $\frac{1}{p}$-security in the presence of aborting strategies.

1.1 Our Results

Our first result concerns with the coin-tossing functionality where we show that it is possible to achieve both simulation against non-aborting adversaries as well as $1/p$-security in the presence of aborting adversaries. More precisely, we prove the following theorem:

Theorem 1.1 (Informal)**.** *Assuming the discrete logarithm problem is hard, there exists a four-round protocol that securely realizes the coin-tossing functionality in the presence of non-aborting adversaries. Moreover, the protocol achieves $1/p$-security in the presence of aborting adversaries.*

We remark that if we allow our simulator to run in expected polynomial-time, we actually obtain perfect simulation against one of the parties and $1/p$-security against the other (even against aborting adversaries). On the other hand, if we require strict polynomial-time simulation, where this polynomial is independent of the adversary's running time, our protocol achieves $1/p$-security relative to both corruption cases. We further provide an abstraction for this protocol using a two-round cryptographic primitive denoted by homomorphic trapdoor commitment scheme, where the commitment transcript, as well as the trapdoor are homomorphic. This abstraction captures additional commitment schemes with security under a larger class of hardness assumptions such as RSA and factoring.

Next, we consider the oblivious-transfer (OT) functionality which is a *complete* primitive for secure two-party computation. We recall that in [ORS15], Ostrovsky, Richelson and Scafuro showed how to construct a four-round oblivious-transfer protocol, while upon combining it with the result of Ishai et al. [IKO^{+}11], they obtained a five-round secure two-party protocol. In this work, we obtain the following theorem:

Theorem 1.2 (Informal)**.** *Assuming the Decisional Diffie-Hellman problem is hard, there exists a four-round oblivious-transfer protocol, where the receiver learns the output in three rounds, that is secure in the presence non-aborting adversaries. Furthermore, our protocol achieves $1/p$-security in the presence of aborting senders.*

Since the receiver learns the output in the third round, we show how to combine this with the protocol of [IKO$^+$11] to obtain a four-round secure two-party computation with analogous security guarantees. Finally, we show that achieving $1/p$-security against aborting receivers is impossible under black-box construction. More formally, we prove the following theorem:

Theorem 1.3 (Informal)**.** *Assuming $\mathsf{NP} \not\subseteq \mathsf{BPP}$, there exists no black-box construction of a three-round secure computation protocol for arbitrary functionalities where only one party receives the output with $1/p$-security against the receiver (of the output).*

Our proof follows by extending the [GK96] lower bound that shows that three-round black-box zero-knowledge proofs (or arguments) with $1/p$-security exist only for languages in BPP. In that sense, our protocols are optimal with respect to the relaxations considered in this work.

1.2 Our Techniques

Coin tossing. We briefly sketch the technical details of our constructions, beginning with our coin tossing protocol. In this protocol we make use of an extension variant of of Pedersen's trapdoor commitment scheme [Ped91]. Basically, party P_1 generates a set of generators for P_2's commitment scheme using pairs of shares, and then reveals the discrete logarithm of half of the shares by responding to a random challenge given by P_2. Looking ahead, this allows to construct a simulator that extracts a trapdoor for this commitment scheme using rewinding which, in turn, allows the equivocation of the committed message. Forcing a particular outcome when P_2 is corrupted is carried out by first observing the decommitted value of P_2 and then rewinding, where in the second execution the simulator programs its input according to the outcome it received from the trusted party. Note that this commitment scheme is captured under our abstraction for trapdoor commitment schemes.

Oblivious transfer. As a warmup, our first OT protocol employs a common paradigm for securely realizing this functionality. Namely, the receiver picks two public keys for which it knows only one of the corresponding secret keys, and sends them to the sender. The sender is next using these keys to encrypt its OT inputs. If indeed the receiver knows only one of the secret keys, then it will not be able to decrypt both inputs. Thus, the main challenge in designing OT protocols with security in the presence of malicious adversaries has always been regarding the way to enforce the receiver to choose its public keys correctly. In this work we enforce that by asking the public key for the unknown secret key to take a particular form, for which the receiver does not know the trapdoor associated with it (concretely, this trapdoor is a discrete logarithm of some generator picked by the sender). Enforcing this choice is carried out by a witness-indistinguishable proof-of-knowledge (WI-PoK), that further allows to extract the bit b for which the receiver indeed knows the corresponding secret key (which implies input extraction of the receiver's input).

On a very high-level, our security guarantee against malicious receivers is achieved by first obtaining a three-round protocol that is defensibly private with respect to malicious receivers [Hai08, HIK$^+$11] and then combining it with a zero-knowledge proof-of-knowledge (ZK-PoK) protocol in order to achieve full security against malicious (non-aborting) adversaries. Loosely speaking, an OT protocol is said to be defensibly-private with respect to the receiver if no adversarial receiver can distinguish the sender's input corresponding to input $1 - b$ from a random input, while outputting a valid defense, i.e. random coins τ that are consistent with the view for input b. Given a defensibly-private OT protocol, obtaining a protocol that guarantees full security against a malicious non-aborting receiver is obtained by combining it with a ZK-PoK protocol where the receiver proves knowledge of a valid defense. (We stress that in our actual protocol, a witness-indistinguishability proof as opposed to a zero-knowledge proof will be sufficient).

In case the sender aborts then we can only ensure privacy with respect to the receiver's inputs. This is because the sender cannot learn anything about the receiver's input, as the simulation is perfect. In order to enhance the security guarantee with respect to the malicious sender we construct another OT protocol, relying on the recent protocol from [ORS15]. More concretely, our first observation is that the previous protocol is already $1/p$-secure for $p = 1 + \frac{1}{3}$. To see this, we first mention that in our warmup protocol the sender picks two trapdoors and the receiver is allowed to choose one of them to be opened by the sender. Specifically, security against malicious receivers still holds since one of the trapdoors remains hidden. Simulation, on the other hand, is achieved by rewinding and extracting both the trapdoors. In case of abort, it is not guaranteed that the simulator can extract both the trapdoors. Nevertheless, we can achieve $1/p = \frac{3}{4}$-security with respect to malicious (possibly aborting) senders. Namely, suppose that for some trapdoor the sender aborts with probability at most $\frac{1}{2}$ when it is asked to reveal it, then in expectation the simulator needs to rewind the sender just 2 times in order to extract that trapdoor. If both trapdoors satisfy this condition then the simulator can easily extract both of them.

Now, suppose this is not the case, then it would have to be the case that the sender aborts with probability at least $\frac{1}{2}$ when it is asked to open one of the trapdoors. Then, the overall probability with which the sender aborts is $\frac{1}{4}$ (as each trapdoor is requested to be revealed with probability $\frac{1}{2}$). In order to achieve $\frac{3}{4}$-security, it suffices to output a distribution that is $\frac{3}{4}$-close to the real distribution. As the sender aborts with probability at least $\frac{1}{4}$ a simulator that simply outputs all the views on which the sender aborts already achieves $\frac{3}{4}$-security. With this observation, we show that in order to get $\frac{1}{p}$-security for an arbitrary polynomial p, we amplify the indistinguishability via parallel repetition. More precisely, by repeating the basic protocol $O(\kappa p)$ times, where κ is the security parameter, we can show that if the adversary does not abort with probability at least $O(\frac{1}{p})$, then the simulation can extract most of the trapdoors. This idea is used in conjunction with the combiner of Ostrovsky, Richelson and Scafuro [ORS15] to ensure that the simulator extracts the sender's inputs if and only if the receiver successfully extracts it, or in other words, prevents any form of input dependent attacks.

1.3 Related Work

Notably relevant to our work is the two-round OT protocol [NP01] developed by Naor and Pinkas that obtains one-sided simulation with respect to the sender, whereas the receiver's security is obtained via an indistinguishability argument. The broader notion of input-indistinguishable computation, introduced by Micali, Pass and Rosen [MPR06], considers a weaker security notion for two-party computation which requires that no party should be able to distinguish two views generated based on distinct set of inputs for the other party but yield the same output. Their main motivation was to develop a meaningful notion of security that supports the design of protocols that can be proven secure in a concurrent setting in the plain model (i.e. assuming no trusted setup). In a later work [HK12], Halevi and Kalai introduced a general framework for two-round OT protocols, considering the weaker security notion of input indistinguishability that does not support input extraction, and design protocols based on smooth projective hashing.

Another weaker security notion is that of covert security, introduced by Aumann and Lindell in [AL10], which models covert adversaries that may deviate arbitrarily from the protocol specification in an attempt to cheat. In their weaker definition, the simulator is allowed to fail, as long as it is guaranteed that the real and ideal output distributions are distinguishable with a probability that is related to the probability of detecting cheating. We note that our security notion directly implies covert security as the simulator may only fail in case the adversary aborts, which is always detected as cheating.

In the context of round optimal secure computation, Faust et al. [DFH12] built a general two-round two-party protocol with polylogarithmic communication in the circuit's size based on extractable hash functions. In a recent work, Ishai et al. [IKKPC15] study two-round secure multi-party computation in an honest majority setting (in the presence of a single corrupted party).

2 Preliminaries

2.1 Basic Notations

We denote the security parameter by n. We say that a function $\mu : \mathbb{N} \to \mathbb{N}$ is *negligible* if for every positive polynomial $p(\cdot)$ and all sufficiently large n it holds that $\mu(n) < \frac{1}{p(n)}$. We use the abbreviation PPT to denote probabilistic polynomial-time. We further denote by $a \leftarrow A$ the random sampling of a from a distribution A, and by $[n]$ the set of elements $\{1, \dots, n\}$.

Computational indistinguishability. We specify the definitions of computational indistinguishability and computational $\frac{1}{p}$-indistinguishability.

Definition 2.1. *Let $X = \{X(a,n)\}_{a\in\{0,1\}^*,n\in\mathbb{N}}$ and $Y = \{Y(a,n)\}_{a\in\{0,1\}^*,n\in\mathbb{N}}$ be two distribution ensembles. We say that X and Y are* computationally indistinguishable*, denoted $X \overset{\rm c}{\approx} Y$, if for every* PPT *distinguisher D there exists a negligible function $\mu(\cdot)$ such that for every $a \in \{0,1\}^*$ and all sufficiently large n*

$$\big|\Pr\left[D(X(a,n),1^n) = 1\right] - \Pr\left[D(Y(a,n),1^n) = 1\right]\big| < \frac{1}{\mu(n)}.$$

Definition 2.2. *Let $X = \{X(a,n)\}_{a\in\{0,1\}^*,n\in\mathbb{N}}$ and $Y = \{Y(a,n)\}_{a\in\{0,1\}^*,n\in\mathbb{N}}$ be two distribution ensembles. We say that X and Y are* computationally $\frac{1}{p}$-indistinguishable*, denoted $X \overset{1/{\rm p}}{\approx} Y$, if for every* PPT *distinguisher D there exists a negligible function $\mu(\cdot)$ such that for every $a \in \{0,1\}^*$ and all sufficiently large n*

$$\big|\Pr\left[D(X(a,n),1^n) = 1\right] - \Pr\left[D(Y(a,n),1^n) = 1\right]\big| < \frac{1}{p(n)} + \frac{1}{\mu(n)}.$$

Statistical distance. Next we specify the distance measure of statistical closeness.

Definition 2.3. *Let X_n and Y_n be random variables accepting values taken from a finite domain $\Omega \subseteq \{0,1\}^n$. The* statistical distance *between X_n and Y_n is*

$$SD(X_n, Y_n) = \frac{1}{2}\sum_{\omega\in\Omega} |\Pr[X_n = \omega] - \Pr[Y_n = \omega]|.$$

We say that X_n and Y_n are ε-close *if their statistical distance is at most $SD(X_n, Y_n) \leq \varepsilon(n)$. We say that X_n and Y_n are* statistically close*, denoted $X_n \approx_s Y_n$, if $\varepsilon(n)$ is negligible in n.*

2.2 Hardness Assumptions

Our constructions rely on the following hardness assumptions.

Discrete logarithm. The classic discrete logarithm assumption is stated as follows.

Definition 2.4 (DL). *We say that* the discrete logarithm (DL) problem is hard relative to $\mathcal{G}$*, if for any PPT adversary A there exists a negligible function* negl *such that*

$$\Pr\left[x \leftarrow A(\mathbb{G}, p, g, g^x)\right] \leq \mathsf{negl}(n),$$

where $(\mathbb{G}, p, g) \leftarrow \mathcal{G}(1^n)$ and the probability is taken over the choice of $x \leftarrow \mathbb{Z}_p$.

Decisional Diffie-Hellman. The decisional Diffie-Hellman assumption is stated as follows.

Definition 2.5 (DDH). *We say that* the decisional Diffie-Hellman (DDH) problem is hard relative to $\mathcal{G}$, *if for any PPT distinguisher D there exists a negligible function* negl *such that*

$$\Big|\Pr\left[D(\mathbb{G},p,g,g^x,g^y,g^z)=1\right]-\Pr\left[D(\mathbb{G},p,g,g^x,g^y,g^{xy})=1\right]\Big|\le \mathsf{negl}(n),$$

where $(\mathbb{G},p,g)\leftarrow\mathcal{G}(1^n)$ and the probabilities are taken over the choices of $x,y,z\leftarrow\mathbb{Z}_p$.

2.3 Commitment Schemes

Statistically hiding commitment schemes maintain two important security properties of hiding and biding, where the flavour of the hiding property is statistical. More formally,

Definition 2.6. *A commitment scheme is a pair of probabilistic polynomial-time algorithms, denoted* $(\mathrm{Sen},\mathrm{Rec})$ *(for sender and receiver), satisfying the following:*

- **Inputs:** *The common input is a security parameter 1^n. The sender has a secret input $m\in\mathcal{M}_n$.*
- **Hiding:** *For every probabilistic polynomial-time algorithms* Rec^* *interacting with* Sen *and every two messages $m,m'\in\mathcal{M}_n$, the random variables describing the output of* Rec^* *in the two cases, namely* $\langle\mathrm{Sen}(m),\mathrm{Rec}^*\rangle(1^n)$ *and* $\langle\mathrm{Sen}(m'),\mathrm{Rec}^*\rangle(1^n)$, *are statistically close.*
- **Binding:** *A receiver's view of an interaction with the sender, denoted $(r,\bar{m})$, consists of the random coins used by the receiver (namely, r) and the sequence of messages received from the receiver (namely, $\bar{m}$).*

  Let $m,m'\in\mathcal{M}_n$. We say that the receiver's view (of such interaction), $(r,\bar{m})$, is a possible m-commitment *if there exists a string s such that $\bar{m}$ describes the messages received by* Rec *when* Rec *uses local coins r and interacts with* Sen *which uses local coins s and has input $(1^n,m)$. We denote $\bar{m}$ by* $\mathbf{View}_{\langle\mathrm{Sen}(m),\mathrm{Rec}\rangle(1^n)}$.

  We say that the receiver's view $(r,\bar{m})$ is ambiguous *if is it both a possible m-commitment and a possible m'-commitment.*

  The binding property asserts that, for all but a negligible fraction of the coins toss of the receiver, there exists no sequence of messages (from the sender) which together with these coin toss forms an ambiguous receiver view. Namely, that for all but a negligible function of the $r\in\{0,1\}^{\mathsf{poly}n}$ there is no $\bar{m}$ such that $(r,\bar{m})$ is ambiguous.

2.3.1 Trapdoor Commitment Schemes

Loosely speaking, a trapdoor commitment scheme is a commitment scheme that meets the classic binding and hiding security properties specified in Definition 2.6, yet it allows to decommit a commitment into any value from the message space given some trapdoor information. In this paper we view the commit phase of the trapdoor commitment schemes as a two-round protocol $\pi_{\mathrm{COM}}=(\pi_{\mathrm{Rec}},\pi_{\mathrm{Sen}})$ where the receiver sends the message π_{Rec} and the sender responds with message π_{Sen} (that is, the receiver knows the trapdoor, where in the simulation, the simulator extracts this trapdoor from the receiver in order to equivocate its commitment). Formally stating,

Definition 2.7. *A two-round trapdoor commitment scheme is a pair of probabilistic polynomial-time algorithms, denoted* $(\mathrm{Sen},\mathrm{Rec})$ *(for sender and receiver), satisfying the following:*

- **Inputs:** *The common input is a security parameter 1^n. The sender has a secret input $m \in \mathcal{M}_n$.*
- $(\mathrm{Sen}, \mathrm{Rec})$ *is a commitment scheme in the sense of Definition 2.6 with perfect hiding.*
- *For any probabilistic polynomial-time algorithm* Rec^* *there exists a polynomial-time algorithm $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ such that for any sequence of messages $\{m_n\}_{n\in\mathbb{N}}$ where $m_n \in \mathcal{M}_n$ for all n, the following holds:*

  On input 1^n simulator $\mathcal{S}_1$ (playing the receiver) outputs π_{Rec} and a trapdoor td.

  Simulator $\mathcal{S}_2$ is defined as follows:

  - *First, on input 1^n and randomness R, $\mathcal{S}$ outputs $\pi^{\mathcal{S}}_{\mathrm{Sen}}$ in response to π_{Rec} such that the distributions of $\{\pi^{\mathcal{S}}_{\mathrm{Sen}}\}_{n\in\mathbb{N}}$ and $\{\pi_{\mathrm{Sen}}\}_{n\in\mathbb{N}}$ are identical.*
  - *Next, on input* td, *message m_n and randomness R, simulator $\mathcal{S}_2$ outputs coins s such that $\pi^{\mathcal{S}}_{\mathrm{Sen}} = \mathrm{Sen}(1^n, \pi_{\mathrm{Rec}}, m_n; s)$.*

Homomorphic trapdoor commitment schemes. We consider trapdoor commitments that are homomorphic in the sense that given two receiver's messages π^1_{Rec} and π^2_{Rec} that are defined relative to some group $\mathbb{G}$, it is possible to combine them into a single receiver's message $\pi_{\mathrm{Rec}} = \pi^1_{\mathrm{Rec}} \cdot \pi^2_{\mathrm{Rec}}$. Moreover, the trapdoor can be homomorphically updated as well. One such example is Pedersen's commitment scheme that is based on the hardness of Discrete logarithm [Ped91]. Loosely speaking, given a group description $\mathbb{G}$ of prime order p, and two generators g, h, a commitment of $m \in \mathbb{Z}_p$ is computed by $c = g^m h^r$ for a random $r \leftarrow \mathbb{Z}_p$. Moreover, the knowledge of $\log_g h$ enables to open c into any message in $\mathbb{Z}_p$. Note that given two generators h_0 and h_1 one can assemble a new generator $h_0 h_1$ for which the trapdoor will be $\log_g h_0 + \log_g h_1$.

Two additional trapdoor commitment schemes that fit to our framework are number-theoretic based constructions in composite order groups. Concretely, we consider two constructions in $\mathbb{Z}^*_N$ for RSA composite N with security based on the RSA and factoring hardness assumptions. Notably, the trapdoor information of these constructions does not require the knowledge of the factorization of N, thus N can be part of the group description handed to the parties at the onset of the protocol (similarly to the group description $\mathbb{G}$ in the prior example). Loosely speaking, a commitment to a message $m \in \mathbb{Z}_e$ in the RSA-based construction is computed by $g^m r^e \bmod N$, where r is picked at random from $\mathbb{Z}^*_N$, $g = x^e \bmod N$ and (N, e) can be considered as the public parameters (such that e is relatively prime to $\varphi(N)$). Moreover, the trapdoor picked by the receiver is x. Clearly, given $g_1 = x_1^e \bmod N$ and $g_2 = x_2^e \bmod N$, then $g_1 g_2 = (x_1 x_2)^e \bmod N$.

An additional factoring-based trapdoor construction implies a commitment to a message $m \in \mathbb{Z}_{2^t}$ by $g^m r^{2^{\tau+t}} \bmod N$ for a random r, such that $g = x^{2^{\tau+t}} \bmod N$ and (N, τ, t) can be considered as the public parameters. Moreover, the trapdoor picked by the receiver is x. The detailed descriptions of these commitment schemes are found in [Fis01].

2.4 Witness Indistinguishability

A proof system between a prove and a verifier is witness indistinguishable if the proof does not leak information about which witness the prover is using, even if the verifier is malicious. In the following, we let $\langle \mathcal{P}(y), \mathcal{V}(z)(x) \rangle$ denote the view of verifier $\mathcal{V}$ when interacting with prover $\mathcal{P}$ on common input x, when $\mathcal{P}$ has auxiliary input y and $\mathcal{V}$ has auxiliary input z.

Definition 2.8. *[FS90] Let $L \in \mathsf{NP}$ and let $(\mathcal{P}, \mathcal{V})$ be an interactive proof system for L with perfect completeness. We say that $(\mathcal{P}, \mathcal{V})$ is* witness-indistinguishable *(WI) if for every* PPT *algorithm $\mathcal{V}^*$ and every two sequences $\{w^1_x\}_{x\in L}$ and $\{w^2_x\}_{x\in L}$ such that w^1_x and w^2_x are both witnesses for $x \in L$, the following ensembles are computationally indistinguishable:*

1. $\{\langle \mathcal{P}(w_x^1), \mathcal{V}(z)\rangle(x)\}_{x\in L, z\in\{0,1\}}$.

2. $\{\langle \mathcal{P}(w_x^2), \mathcal{V}(z)\rangle(x)\}_{x\in L, z\in\{0,1\}}$.

2.5 Secret-Sharing

A secret-sharing scheme allows distribution of a secret among a group of n players, each of whom in a *sharing phase* receive a share (or piece) of the secret. In its simplest form, the goal of secret-sharing is to allow only subsets of players of size at least $t+1$ to reconstruct the secret. More formally a $t+1$-out-of-n secret sharing scheme comes with a sharing algorithm that on input a secret s outputs n shares $s_1, \ldots, s_n$ and a reconstruction algorithm that takes as input $(s_i)_{i\in S}, S$ where $|S| > t$ and outputs either a secret s' or $\perp$. In this work, we will use the Shamir's secret sharing scheme [Sha79] with secrets in $\mathbb{F} = GF(2^\kappa)$. We present the sharing and reconstruction algorithms below:

Sharing algorithm: For any input $s \in \mathbb{F}$, pick a random polynomial $f(\cdot)$ of degree t in the polynomial-field $\mathbb{F}[x]$ with the condition that $f(0) = s$ and output $f(1), \ldots, f(n)$.

Reconstruction algorithm: For any input $(s_i')_{i\in S}$ where none of the s_i' are $\perp$ and $|S| > t$, compute a polynomial $g(x)$ such that $g(i) = s_i'$ for every $i \in S$. This is possible using Lagrange interpolation where g is given by

$$g(x) = \sum_{i\in S} s_i' \prod_{j\in S/\{i\}} \frac{x-j}{i-j} \ .$$

Finally the reconstruction algorithm outputs $g(0)$.

We will additionally rely on a property of this secret-sharing scheme that has been observed by Ostrovsky, Richelson and Scafuro in [ORS15]. Towards that, we view the Shamir secret-sharing scheme as a linear code generated by the following $n \times (t+1)$ Vandermonde matrix

$$A = \begin{pmatrix} 1 & 1^2 & \cdots & 1^t \\ 1 & 2^2 & \cdots & 2^t \\ \vdots & \vdots & \vdots & \vdots \\ 1 & n^2 & \cdots & n^t \end{pmatrix}$$

More formally, the shares of a secret s that are obtained via a polynomial f in the Shamir scheme, can be obtained by computing $A\mathbf{c}$ where $\mathbf{c}$ is the vector containing the coefficients of f. Next, we recall that for any linear code A, there exists a parity check matrix H of dimension $(n-t-1)\times n$ which satisfies the equation $HA = \mathbf{0}_{(n-t-1)\times(t+1)}$, i.e. the all 0's matrix. We thus define the linear operator $\phi(v) = Hv$ for any vector v. Then it holds that any set of shares $\mathbf{s}$ is valid if and only if it satisfies the equation $\phi(\mathbf{s}) = \mathbf{0}_{n-t-1}$.

3 A 4-Round Coin Tossing Protocol from Discrete Logarithm

In this section we present a four-round coin tossing protocol that is based on the hardness of the discrete logarithm problem. Namely, the parties use an extension of Pedersen's trapdoor commitment scheme [Ped91] that is based on n generators. Basically, party P_1 generates the generators for P_2's commitment scheme using pairs of shares, and then reveals the discrete logarithm of half of the shares by responding to a random challenge given by P_2. Looking ahead, this allows to construct a simulator that extracts a trapdoor for this commitment scheme using rewinding which, in turn, allows the equivocation of the committed message. Forcing a particular outcome when P_2 is corrupted is carried out by first observing the decommitted value of P_2 and then rewinding, where in the second execution the simulator programs its input according to the outcome it received from the trusted party.

We note that for both corruption cases, we construct *universal* simulators (namely, simulators that do not depend on the code of the adversary), that run in *strict* polynomial-time and induce $\frac{1}{p}$-security. The simulator for a corrupted P_1 can be modified into an expected time simulator with full security as in the usual sense. The security of P_2 cannot be further enhanced as it learns the coin tossing outcome after in the third round, and may choose to abort right after. Essentially, the problem in acute when the adversary's non-aborting probability in the last message is negligible, as it prevents from generating a view that is consistent with the coin-tossing outcome even using rewinding. Conditioned on this event, we prove that the difference between the simulated and real views is at most $1/p(n)$.

We are now ready to present our protocol in details.

Protocol 1 (Protocol π_{COIN}).

Public parameters: *The description of a group $\mathbb{G}$ of prime order p and a generator g.*

The protocol:

1. $\mathbf{P_1} \rightarrow \mathbf{P_2}$*: Pick random elements $r_0^1, r_1^1, \ldots, r_0^n, r_1^n \leftarrow \mathbb{Z}_p$ and sends P_2 the pairs $(h_0^1, h_1^1), \ldots, (h_0^n, h_1^n)$, where $h_b^i = g^{r_0^i}$ for all $b \in \{0,1\}$ and $i \in [n]$.*
2. $\mathbf{P_2} \rightarrow \mathbf{P_1}$*: Pick random elements $m, s_1, \ldots, s_n \leftarrow \mathbb{Z}_p$ and compute*
$$\sigma = g^m (h_0^1 h_1^1)^{s_1} \cdots (h_0^n h_1^n)^{s_n}.$$
Select random bits $e_1, \ldots, e_n$ and send $\sigma, e_1, \ldots, e_n$ to P_1.
3. $\mathbf{P_1} \rightarrow \mathbf{P_2}$*: Pick a random $m' \leftarrow \mathbb{Z}_p$ and send $m', r_{e_1}^1, \ldots, r_{e_n}^n$ to P_2.*
4. $\mathbf{P_2} \rightarrow \mathbf{P_1}$*: Compute the coin tossing outcome as $m + m' \bmod p$ and send $m, s_1, \ldots, s_n$ to P_1.*

Theorem 3.1. *Assume that the discrete logarithm assumption holds in $\mathbb{G}$. Then Protocol 2 securely computes $\mathcal{F}_{\text{COIN}}$ in the presence of static, malicious adversaries (in the sense of Definition A.4 with $\frac{1}{p}$ security).*

Proof: We consider each corruption case separately.

P_1 **is corrupted.** On a high-level, in order to simulate P_1 we construct a simulator $\mathcal{S}$ that extracts the trapdoor for one of the pairs h_0^i, h_1^i sent in the first message, namely, the discrete logarithm of both elements in the pair with respect to g, and then uses that to equivocate P_2's commitment in the last message. More precisely, for any probabilistic polynomial-time adversary $\mathcal{A}$ controlling P_1 we define a simulator $\mathcal{S}$ that is given an input m_o from $\mathcal{F}_{\text{COIN}}$ and proceeds as follows:

1. $\mathcal{S}$ internally invokes $\mathcal{A}$. Upon receiving the first message from $\mathcal{A}$, it feeds $\mathcal{A}$ with a second message generated using the honest P_2's strategy. Let $\sigma, e_1, \ldots, e_n$ be the message fed to $\mathcal{A}$ and $m, s_1, \ldots, s_n$ be the randomness used to generate the forth message (which is determined by the second message).

2. If $\mathcal{A}$ aborts before providing the third message, $\mathcal{S}$ halts outputting $\perp$. If $\mathcal{A}$ provides a third message, then $\mathcal{S}$ stalls the main execution and proceeds to rewind $\mathcal{A}$. Specifically, $\mathcal{S}$ rewinds $\mathcal{A}$ to the second message and supplies a different second message by sampling uniformly random coins for the honest P_2's strategy. Let $\tilde{e}_1, \ldots, \tilde{e}_n$ be the bits sent within the rewinded second message. If $\mathcal{A}$ responds, then $\mathcal{S}$ finds an index j such that $\tilde{e}_j \neq e_j$. Note that such an index j implies that $\mathcal{S}$ now has t_0 and t_1 such that $h_0^j = g^{t_0}$ and $h_1^i = g^{t_1}$. Else, if $\mathcal{A}$ aborts then $\mathcal{S}$ rewinds $\mathcal{A}$ to the second message and tries another freshly generated second message. $\mathcal{S}$ repeats this procedure $np(n)$ times and outputs fail if (1) the challenges $\tilde{e}_1, \ldots, \tilde{e}_n$ are identical to $e_1, \ldots, e_n$ in any of the attempts or, (2) in case all the attempts were unsuccessful.

3. Finally, $\mathcal{S}$ proceeds to complete the main execution conditioned on not outputting fail. Let m' be part of the third message supplied by $\mathcal{A}$ and let σ be the message fed to $\mathcal{A}$ as part of the second message. $\mathcal{S}$ computes
$$\tilde{s}_j = (m - m_o + m' + (t_0 + t_1)s_j)/(t_0 + t_1) \bmod p$$
and for all other i, $\tilde{s}_i = s_i$. As the final message $\mathcal{S}$ feeds $\mathcal{A}$ with $(m_o - m'), \tilde{s}_1, \ldots, \tilde{s}_n$.

We first argue for the correctness of the simulation. This follows from the ability to equivocate the commitment employed by P_2 once the discrete logarithm of one of the $h_0^i h_1^i$ elements is known to the simulator. More formally, let j be as in the simulation for which the simulator obtains t_0 and t_1 such that $h_0^j = g^{t_0}$ and $h_1^i = g^{t_1}$. Moreover, let $\sigma = g^m (h_0^1 h_1^1)^{s_1} \cdots (h_0^n h_1^n)^{s_n}$ as computed by the simulator in the second message of the simulation (note that σ is fixed once for the entire simulation and is never modified). We focus our attention on the product $g^m (h_0^j h_1^j)^{s_j}$, where s_j is the randomness revealed by the simulator in the third message. An important observation here is that it is sufficient to equivocate this product in order to equivocate the entire commitment. Namely, if the simulator can come up with two distinct pairs (m, s_j) and $(\tilde{m}, \tilde{s}_j)$ such that $g^m (h_0^j h_1^j)^{s_j} = g^{\tilde{m}} (h_0^j h_1^j)^{\tilde{s}_j}$, then it is possible to conclude two distinct openings with respect to the commitment used by P_2 by reusing the same $\{s_i\}_{i \neq j}$. Finally, since the simulator obtains t_0 and t_1 as above, it can conclude the discrete logarithm of $h_0^j h_1^j$ relative to g which corresponds to $t_0 + t_1$. Putting it all together, the simulator can easily equivocate $\ell = g^m (h_0^j h_1^j)^{s_j}$ into the message $m_o - m'$ (which will imply that the two shares yield m_o), by computing $\tilde{s}_j$ as follows. Consider the linear equation implied in the exponent of ℓ which equals $m + (t_0 + t_1)s_j$, then $m + (t_0 + t_1)s_j = m_o - m' + (t_0 + t_1)\tilde{s}_j$, which implies that $\tilde{s}_j = (m - m_o + m' + (t_0 + t_1)s_j)/(t_0 + t_1) \bmod p$. Next we prove that,

Claim 3.1. *There exists a negligible function* $\mathsf{negl}(\cdot)$ *for which* $\mathcal{S}$ *outputs* fail *with probability at most* $\frac{1}{p(n)} + \mathsf{negl}(n)$.

Proof: First, we consider a hybrid simulator $\tilde{\mathcal{S}}$ that instead of rewinding only $np(n)$ times, repeatedly rewinds until it successfully obtains two responses from $\mathcal{A}$ relative to the third message. Moreover, $\tilde{\mathcal{S}}$ does not abort if the same challenge message occurs for a second time. We will next argue that the expected running time of $\tilde{\mathcal{S}}$ is polynomial. Let ε denote the probability that $\mathcal{A}$ answers correctly on the third message. We consider two cases: (1) $\mathcal{A}$ aborts in the first simulated run (which occurs with probability $1 - \varepsilon$). In this case the simulator outputs $\perp$. (2) $\mathcal{A}$ does not abort in the first simulated run (which occurs with probability ε). In this case the expected number of rewinding attempts $\tilde{\mathcal{S}}$ performs before $\mathcal{A}$ provides another valid third message is $\frac{1}{\varepsilon}$. Therefore, the expected number of times of $\tilde{\mathcal{S}}$ rewinds $\mathcal{A}$ is
$$(1 - \varepsilon) + \varepsilon \frac{1}{\varepsilon} = O(1).$$
Next, we bound the probability of the strict simulator $\mathcal{S}$ outputting fail by computing the probability that it outputs fail in each of the cases. (1) The probability that $\mathcal{A}$ does not provide a third message within the $np(n)$ attempts can be bounded using the Markov inequality, as the probability that $\tilde{\mathcal{S}}$ carries out more than $np(n)$ rewinding attempts is at most $\frac{O(1)}{np(n)} < \frac{1}{2p(n)}$. (2) Next, the probability that $\mathcal{S}$ fails due to the event that the same challenge occurred twice can be bounded using a union bound argument which yields a value bounded by $np(n) \times \frac{1}{2^n}$. We conclude that the overall probability that $\mathcal{S}$ outputs fail is bounded by $\frac{1}{2p(n)} + \frac{np(n)}{2^n} < \frac{1}{p(n)}$. $\square$

Claim 3.2. *The following two distribution ensembles are computationally* $\frac{1}{p(n)}$*-indistinguishable,*
$$\left\{\mathbf{View}_{\pi_{\mathrm{COIN}},\mathcal{A}(z)}(n)\right\}_{n \in \mathbb{N}, z \in \{0,1\}^*} \overset{1/\mathrm{p}}{\approx} \left\{\mathbf{View}_{\mathcal{F}_{\mathrm{COIN}},\mathcal{S}(z)}(n)\right\}_{n \in \mathbb{N}, z \in \{0,1\}^*}.$$

Proof: Finally, we wish to claim that the adversary's view in both real and simulated executions is identically distributed conditioned on the event that $\mathcal{S}$ does not output fail or abort. Note that the adversary's view is comprised from $\sigma, e_1, \ldots, e_n$ in the second message, and $m_o - m', s_1, \ldots, s_{j-1}, \tilde{s}_j, s_{j+1}, \ldots, s_n$ in the fourth message. Moreover, the second message is generated as in the real execution (and thus is distributed identically to the corresponding message in the real execution), whereas the fourth message is generated by first producing a real execution message and then equivocating the outcome commitment. We claim that the fourth simulated message is identically distributed to the fourth real message. On a high-level, this is due to the fact that m_o and m' are picked uniformly at random by $\mathcal{F}_{\text{COIN}}$ and $\mathcal{S}$, respectively, and so $m_o - m'$ is a uniformly distributed element in $\mathbb{Z}_p$. Moreover, $\tilde{s}_j$ depends on the distribution of s_j which is uniformly random in $\mathbb{Z}_p$ as well.

More formally, our construction implies that the real and simulated views are indistinguishable relative to the partial views where the adversary aborts before sending the third message. It is therefore suffices to show that the adversary's views are indistinguishable conditioned on not aborting in the simulation. More precisely, we prove that the distribution of $m_o - m', \tilde{s}_1, \ldots, \tilde{s}_n$ in the simulated view is identically distributed to the real view conditioned on m_o being the outcome of the coin tossing functionality, m' being the adversary's share, σ being the second message and the adversary not aborting in the third message. It follows from our simulation that the distributions of $\tilde{s}_i$ for $i \neq j$ are identical as in both executions these values are sampled uniformly. Now, given that these values are already fixed, there exist unique values m and $\tilde{s}_j$ that can be sent as part of the fourth message, which yield a consistent view with m_o. Hence, the views are identically distributed.

From Claim 3.1 we know that the probability $\mathcal{S}$ aborts is at most $\frac{1}{p(n)} + \mathsf{negl}(n)$. Therefore,

$$\Pr[\mathbf{View}_{\mathcal{F}_{\text{COIN}},\mathcal{S}(z)}(n) \neq \perp] \geq 1 - \frac{1}{p(n)} - \mathsf{negl}(n).$$

Combining this claim with the fact that the simulated non-aborted view is identical to the real view, we obtain for every PPT distinguisher D there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all sufficiently large n

$$\left|\Pr\left[D(\mathbf{View}_{\mathcal{F}_{\text{COIN}},\mathcal{S}(z)}(n)) = 1\right] - \Pr\left[D(\mathbf{View}_{\pi_{\text{COIN}},\mathcal{A}(z)}(n)) = 1\right]\right| < \frac{1}{p(n)} + \frac{1}{\mathsf{negl}(n)}.$$

□

P_2 **is corrupted.** Informally, in case P_2 is corrupted the simulator extracts the committed message from $\mathcal{A}$ and then provides a share in the third message that is consistent with m_o and $\mathcal{A}$'s share. More precisely, for any probabilistic polynomial-time adversary $\mathcal{A}$ controlling P_2 we define a simulator $\mathcal{S}$ that is given an input m_o from $\mathcal{F}_{\text{COIN}}$ and proceeds as follows:

1. $\mathcal{S}$ internally invokes $\mathcal{A}$ and computes the first message of the protocol as would have computed by the honest P_1. Namely, $\mathcal{S}$ picks random elements $r_0^1, r_1^1, \ldots, r_0^n, r_1^n \leftarrow \mathbb{Z}_p$ and sends $\mathcal{A}$ the pairs $(h_0^1, h_1^1), \ldots, (h_0^n, h_1^n)$, where $h_b^i = g^{r_0^i}$ for every $b \in \{0,1\}$ and $i \in [n]$. Let $\sigma, e_1, \ldots, e_n$ be the message replied by $\mathcal{A}$.

2. Next, $\mathcal{S}$ performs the following $np(n)$ times:

   - $\mathcal{S}$ picks a random $m' \leftarrow \mathbb{Z}_p$ and sends $m', r_{e_1}^1, \ldots, r_{e_n}^n$ to P_2.

   If at any iteration $\mathcal{A}$ provides a valid fourth message $m, s_1, \ldots, s_n$, then $\mathcal{S}$ rewinds $\mathcal{A}$ to the third message. Next, upon receiving m_o from the ideal functionality, $\mathcal{S}$ supplies $\mathcal{A}$ with a third message $m_o - m, r_{e_1}^1, \ldots, r_{e_n}^n$ and completes the execution. If $\mathcal{A}$ aborts in all the $np(n)$ attempts, $\mathcal{S}$ simply outputs the transcript from the first iteration.

We first prove that if the discrete logarithm assumption is hard in $\mathbb{G}$ then $\mathcal{A}$ cannot open σ in two different valid ways as it violates this hardness assumption.

Claim 3.3. *Assume that the discrete logarithm assumption holds in $\mathbb{G}$. Then, except with negligible probability, $\mathcal{A}$ cannot provide two tuples $m_1, s_1^1, \ldots, s_n^1$ and $m_2, s_1^2, \ldots, s_n^2$ for which $m_1 \neq m_2$, that correspond to valid openings of σ.*

Proof: Assume for contradiction that there exists an adversary $\mathcal{A}$ that can provide two valid distinct decommitments in the fourth round of the protocol with non-negligible probability. We show how to construct an adversary $\mathcal{B}$ that violates the discrete logarithm assumption relative to $\mathbb{G}$. On a high-level, upon given input (g', h'), $\mathcal{B}$ sets $g = g'$ and picks all (h_0^i, h_1^i) pairs honestly with the exception that $h_b^j = h'$ for a randomly chosen $b \in \{0,1\}$ and $j \in [n]$. Next, given two openings $m_1, s_1^1, \ldots, s_n^1$ and $m_2, s_1^2, \ldots, s_n^2$, $\mathcal{B}$ computes the discrete logarithm of h' with respect to $g = g'$. More precisely, denote by t_b^i the discrete logarithm of h_b^i with respect to g for all $b \in \{0,1\}$ and $i \in [n]$, i.e., $h_b^i = g^{t_b^i}$. Then it must hold that

$$m_1 + (t_0^1 + t_1^1)s_1^1 + \ldots + (t_0^n + t_1^n)s_n^1 = m_2 + (t_0^1 + t_1^1)s_1^2 + \ldots + (t_0^n + t_1^n)s_n^2$$

as $\mathcal{A}$ provides two openings to the same commitment σ. Therefore, it is simple to compute

$$t_b^j = \left[m_1 - m_2 + \sum_{i \neq j}(t_0^i + t_1^i)(s_i^1 - s_i^2) + t_{1-b}^j(s_j^1 - s_j^2)\right] / (s_j^2 - s_j^1)$$

which implies that $\mathcal{B}$ violates the discrete logarithm assumption relative to $\mathbb{G}$. □

Claim 3.4. *The following two distribution ensembles are computationally $\frac{1}{p}$-indistinguishable,*

$$\left\{\mathbf{View}_{\pi_{\text{COIN}},\mathcal{A}(z)}(n)\right\}_{n\in\mathbb{N},z\in\{0,1\}^*} \overset{1/\text{p}}{\approx} \left\{\mathbf{View}_{\mathcal{F}_{\text{COIN}},\mathcal{S}(z)}(n)\right\}_{n\in\mathbb{N},z\in\{0,1\}^*}.$$

Proof: Let q be the probability of which $\mathcal{A}$ sends the fourth message. We consider two cases:

Case $q > \frac{1}{p(n)}$: In this case, the probability that $\mathcal{S}$ fails to extract m within the $np(n)$ trials is negligible in n. Moreover, it is easy to argue that whenever $\mathcal{S}$ extracts m, then the distribution generated by $\mathcal{S}$ is identically distributed to the real view conditioned on the adversary not equivocating. Specifically, as this event only occurs with negligible probability (as shown in Claim 3.3), the real and ideal views are statistically close.

Case $q < \frac{1}{p(n)}$: In this case, let t be the probability that $\mathcal{S}$ fails to extract m within the $np(n)$ trials. Let D_a be the distribution of the real view of the adversary conditioned on it aborting in the fourth step, and let D_b be the real view conditioned on the adversary not aborting. Then we can express the distribution of $\mathcal{A}$'s real view as a mixture of distributions as follows:[2]

$$(1-q)D_a + qD_b.$$

The simulator on the other hand will generate a distribution as follows:

$$(1-t)D_a + t((1-q)D_a + qD_b).$$

Then the statistical distance between the two distributions can be computed as the difference

$$||(q - tq)D_a + (tq - q)D_b||_1 = q(1-t)||(D_a - D_b)||_1$$

which is bounded from above by $q < \frac{1}{p(n)}$. Hence the real and simulated view are $\frac{1}{p}$-indistinguishable.

□ □

[2]More precisely, the real view can be obtained by first selecting D_a with probability q and D_b otherwise, and then the selecting a random view in the particular distribution.

3.1 An Abstraction using Homomorphic Trapdoor Commitment Schemes

We further demonstrate how to abstract the protocol from Section 3 based on an homomorphic two-round trapdoor commitment scheme (cf. Section 2.3.1), denoted by $\pi_{\mathrm{COM}} = (\pi_{\mathrm{Sen}}, \pi_{\mathrm{Rec}})$.

Protocol 2 (Protocol π_{COIN})**.**

The protocol:

1. *P_1 (playing the role of the receiver) generates $2n$ pairs of instances of the first message in π_{COM} denoted by $((\pi^0_{\mathrm{Rec}_1}, \pi^1_{\mathrm{Rec}_1}), \ldots, (\pi^0_{\mathrm{Rec}_n}, \pi^1_{\mathrm{Rec}_n}))$ (with independent fresh randomness), and sends these pairs to P_2.*
2. *For all $j \in [n]$, P_2 first combines each pair $(\pi^0_{\mathrm{Rec}_j}, \pi^1_{\mathrm{Rec}_j})$ into a single instance $\tilde{\pi}_{\mathrm{Rec}_j}$ (relying on the homomorphic property of π_{COM}). Next, it shares its coin tossing share m_2 into n shares $m_2^1, \ldots, m_2^n$ and commits to these shares by computing the response to $\tilde{\pi}_{\mathrm{Rec}_j}$, denote these responses by $(\pi_{\mathrm{Sen}_1}, \ldots, \pi_{\mathrm{Sen}_n})$. P_2 additionally sends a random challenge $e \leftarrow \{0,1\}^n$.*
3. *Let $e = (e_1, \ldots, e_n)$. Then P_1 reveals the randomness it used for computing $\pi^{e_j}_{\mathrm{Rec}_j}$ for all $j \in [n]$, and further sends its coin tossing share m_1.*
4. *P_2 verifies that P_1 generated the first message correctly with respect to challenge e. If all the verifications are accepting P_2 opens its commitments from Step 2 and P_1 verifies the validity of this opening. If all the verifications are accepting the parties output $m_1 + m_2$ (where addition is computed in the corresponding group). Otherwise, P_1 aborts.*

Intuitively speaking, Protocol 2 is proven similarly to the proof of Protocol 1. Namely, when P_1 is corrupted the simulator extracts one of the trapdoor pairs of the commitment scheme that enables to equivocate the corresponding receiver's share. On the other hand, when P_2 is corrupted, then the simulator behaves identically to the simulator of P_2 for Protocol 1. That is, the simulator extracts the committed message from the adversary and then rewinds it, providing a new third message that is consistent with m_o. It is simple to verify that Protocol 2 abstracts Protocol 1, that is formally described above in Section 3. Two additional constructions with security under the RSA and the factoring hardness assumptions are captured by our abstraction as well; see Section 2.3.1 for more details.

4 A Warmup: Non-Aborting 4-Round Two-Party Computation

In this section we present our protocol that securely computes functionality $\mathcal{F}_{\mathrm{OT}} : ((s_0, s_1), b) \mapsto (-, s_b)$ in the plain model in the presence of malicious attacks. Our construction implies a four-round oblivious transfer protocol that further induces a four-round two-party protocol with the following security guarantee. Namely, the receiver learns its output already at the third round and the last round is needed in order to extract its input. Namely, in case either party does not abort then we can realize $\mathcal{F}_{\mathrm{OT}}$ under the standard simulation based security. On the other hand, in case the sender aborts, it cannot deduce any information about b since the receiver's input is information theoretically hidden. Finally, in case the receiver aborts, our guarantee is a defensible private OT with respect to malicious receivers ([HIK$^+$11]). We begin by introducing our OT protocol and then explain how to extend it into general two-party computation.

4.1 Building Blocks

Our protocol relies on the following cryptographic building blocks:

Proof of validity. The receiver in our protocol uses a standard Σ-protocol WI-PoK for proving the knowledge of the discrete logarithm of one of the public keys it forwards the sender. The protocol ensures that there is at least one public key for which the receiver knows the discrete logarithm relative to some generator (where this corresponds to the public key for which the receiver does not know the secret key). Concretely, we consider a Σ-protocol $\pi_{\text{DL}}^{\text{WI}}$ for the following language [CEvdG87],

$$\mathcal{L}_{\text{DL}} = \{(g, h, \mathbb{G}, p) | \ \exists u \in \mathbb{Z}_p \text{ such that } h = g^u\}.$$

We note that this proof is given for compound statements. Namely, the parties hold two statements for which the prover only knows one of the witnesses, but not both. It is a common technique by now to combine two Σ-protocols (even distinct ones) in a way that ensures that the prover knows at least one of the witnesses [CDS94]. We note that the compound proof implies a perfect WI-PoK (namely, the view that is produced with respect to one witness is identical to a view that is produced with respect to the other witness). Consequently, even an unbounded verifier cannot tell which witness is used by the prover for proving the compound statement.

The El-Gamal PKE [Gam85] (see Appendix A.1.1). Intuitively speaking, the receiver chooses group elements that will be later viewed by the sender as El Gamal public keys. The key point is that the receiver must pick these elements in two distinct ways, which will be verified by the sender using the WI-PoK $\pi_{\text{DL}}^{\text{WI}}$. If indeed the receiver completes this proof correctly, then we can prove that there exists a public key for which the receiver does not know the trapdoor secret key. This will allow us to claim the privacy of one of the sender's inputs. On the other hand, if the receiver cheats then it may learn both of the sender's inputs. Nevertheless, in this case it will always be caught.

4.2 Our OT Protocol

We are now ready to describe our protocol for secure computation. We construct a four-round OT protocol that guarantees full security assuming that the adversary does not abort. Next, we show how to combine our OT protocol with the [IKO$^+$11] protocol that requires two rounds of communication in the $\mathcal{F}_{\text{OT}}$-hybrid model. That would imply secure two-party computation in four-round with full security in the plain model whenever the adversary does not abort.

Protocol 3 (Protocol π_{OT})**.**

Public parameters: *The description of a group $\mathbb{G}$ of prime order p.*

Inputs: *The sender* Sen *holds s_0, s_1 and the receiver* Rec *holds a bit b.*

The protocol:

1. $\textbf{Sen} \rightarrow \textbf{Rec}:$

   (a) Sen *picks a random generator $g \leftarrow \mathbb{G}$ and computes $h_0 = g^{r_0}$ and $h_1 = g^{r_1}$ where $r_0, r_1 \leftarrow \mathbb{Z}_p$.*

   (b) Sen *sends g, h_0, h_1 to* Rec.

2. $\textbf{Rec} \rightarrow \textbf{Sen}:$

   (a) Rec *generates two public-keys according to the El Gamal PKE as follows:* $\text{PK}_b = g^m$ *and* $\text{PK}_{1-b} = (h_0 h_1)^{\widetilde{m}}$ *where $m, \widetilde{m} \leftarrow \mathbb{Z}_p$.* Rec *sets* $\text{SK} = m$.

   (b) Rec *sends* PK_0, PK_1 *to* Sen.

   (c) Rec *sends the first message of the WI-PoK for proving the knowledge of the discrete logarithms of either* PK_0 *or* PK_1 *with respect to $(h_0 h_1)$ (namely,* Rec *sends the first message with respect to $\pi_{\text{DL}}^{\text{WI}}$ for the compound statement with* PK_0 *and* PK_1 *being the statements).*

   (d) Rec *sends a challenge bit β.*

3. $\textbf{Sen} \rightarrow \textbf{Rec}:$

   (a) Sen *computes ciphertexts c_0, c_1 as follows: $c_0 = (g^{u_0}, \text{PK}_0^{u_0} \cdot s_0)$ and $c_1 = (g^{u_1}, \text{PK}_1^{u_1} \cdot s_1)$ where $u_0, u_1 \leftarrow \mathbb{Z}_p$.*

   (b) Sen *sends* c_0, c_1 *to* Rec
   (c) Sen *sends the second message* e_{Sen} *for the WI-PoK protocol* $\pi_{\mathrm{DL}}^{\mathrm{WI}}$ *given by the receiver (recall that this message is a random challenge).*
   (d) Sen *sends* $r_\beta = \log_g(h_\beta)$

4. **Rec $\rightarrow$ Sen :**
   (a) *Upon receiving the sender's ciphertexts* $c_0 = \langle c_0[1], c_0[2]\rangle$ *and* $c_1 = \langle c_1[1], c_1[2]\rangle$, Rec *computes* s_b *by decrypting* c_b *under* SK_b*. More precisely, it computes* $s_b = c_b[2]/(c_b[1])^{\mathrm{SK}}$.
   (b) Rec *sends the last message for the WI-PoK protocol* $\pi_{\mathrm{DL}}^{\mathrm{WI}}$.

Theorem 4.1. *Assume that the Decisional Diffie-Hellman assumption holds in* $\mathbb{G}$ *and that* $\pi_{\mathrm{DL}}^{\mathrm{WI}}$ *is as above. Then, Protocol 3 securely realizes* $\mathcal{F}_{\mathrm{OT}}$ *in the presence of* non-aborting *static, malicious adversaries.*

Proof overview. First, in case the sender is corrupted the simulator plays the role of the honest receiver with input $b = 0$ and extracts both r_0 and r_1. Next, the simulator uses these values in order to equivocate the public keys it sends to the adversary in the second message. Namely, upon extracting the discrete logarithms of both h_0 and h_1 the simulator knows the secret keys for both public keys and can decrypt both ciphertexts.

On the other hand, in case the receiver is corrupted, security is proven via a reduction to the IND-CPA security game of the El Gamal PKE. Namely, the simulator first extracts the receiver's secret exponent $\widetilde{m}$ and the bit b (from the WI-PoK $\pi_{\mathrm{DL}}^{\mathrm{WI}}$), and uses that information to complete the IND-CPA reduction by plugging in an external public key instead of $(h_0h_1)^{\widetilde{m}}$ and a ciphertext that either encrypts s_{1-b} or a random independent message.

Correctness. On a high-level, correctness follows from the correctness of the El Gamal PKE. Namely, given that the receiver knows the secret key m for PK_b, it can decrypt ciphertext c_b.

Proof: We consider each corruption case separately.

Sen **is corrupted.** Recall that when the sender is corrupted we need to prove that it cannot learn anything about the bit b while extracting both s_0 and s_1. More precisely, consider any probabilistic polynomial-time adversary $\mathcal{A}$ controlling Sen. We define a simulator $\mathcal{S}$ that proceeds as follows:

1. $\mathcal{S}$ invokes $\mathcal{A}$ on its input and randomness of appropriate length.
2. Upon receiving from $\mathcal{A}$ the first message, $\mathcal{S}$ computes the second message honestly with input $b = 0$.
3. Upon receiving $\mathcal{A}$'s third message, $\mathcal{S}$ records r_β. Next, it stalls the main execution and proceeds to rewind $\mathcal{A}$. Specifically, $\mathcal{S}$ rewinds $\mathcal{A}$ to the second message and supplies a bit $1-\beta$. Upon receiving $r_{1-\beta}$, $\mathcal{S}$ completes the main execution honestly using $b = 0$ and decrypts both ciphertexts as follows. $\mathcal{S}$ uses $\mathrm{SK}_0 = \mathrm{SK}$ to decrypt c_0 as the honest receiver would do. Moreover, $\mathcal{S}$ fixes $\mathrm{SK}_1 = (r_0+r_1)\widetilde{m}$ and uses SK_1 to decrypt c_1.
4. Finally, $\mathcal{S}$ forwards (s_0, s_1) to $\mathcal{F}_{\mathrm{OT}}$ and halts, outputting whatever $\mathcal{A}$ does.

Clearly, $\mathcal{S}$ runs in strict polynomial-time. We first prove the correctness of simulation. Specifically, we need to prove that the simulator correctly extracts s_1. Recall that for $b = 1$ the honest receiver computes $s_1 = c_1[2]/(c_1[1])^{\mathrm{SK}}$. Then we claim that this is equivalent to the computation carried out by the simulator, as SK amounts in this case to the discrete logarithm of PK_1 relative to generator g. Next, we prove that,

Claim 4.1. *The following two distribution ensembles are identical,*

$$\left\{\mathbf{View}_{\pi_{\mathrm{OT}},\mathcal{A}(z)}(n,(s_0,s_1),b)\right\}_{n\in\mathbb{N},s_0,s_1,b,z\in\{0,1\}^*} \equiv \left\{\mathbf{View}_{\mathcal{F}_{\mathrm{OT}},\mathcal{S}(z)}(n,(s_0,s_1),b)\right\}_{n\in\mathbb{N},s_0,s_1,b,z\in\{0,1\}^*}.$$

Proof: The proof follows due to the fact that the receiver's bit b is information theoretically hidden given PK_0, PK_1 and the WI-PoK transcript of $\pi_{\mathrm{DL}}^{\mathrm{WI}}$. More concretely, given any pair $(\mathrm{PK}_0, \mathrm{PK}_1)$ there always exist $m^0, \widetilde{m}^0$ and $m^1, \widetilde{m}^1$ for which $\mathrm{PK}_0 = g^{m_0} = (h_0 h_1)^{\widetilde{m}_0}$ and $\mathrm{PK}_1 = g^{m_1} = (h_0 h_1)^{\widetilde{m}_1}$. Moreover, the WI-PoK $\pi_{\mathrm{DL}}^{\mathrm{WI}}$ is a perfect witness indistinguishable proof, which implies that even an unbounded verifier cannot extract b (as discussed above, this is the issue even for the compound proof, since the receiver proves that it knows a discrete logarithm relative to either PK_0 or PK_1). $\square$

Rec **is corrupted.** In this case we need to prove that the corrupted receiver cannot learn anything about the sender's other input s_{1-b} while extracting b. More precisely, for any probabilistic polynomial-time adversary $\mathcal{A}$ controlling Sen we define a simulator $\mathcal{S}$ that proceeds as follows:

1. $\mathcal{S}$ invokes $\mathcal{A}$ on its input and randomness of the appropriate length.
2. $\mathcal{S}$ plays the role of the honest sender with arbitrary inputs (s'_0, s'_1). Upon completing the execution successfully, $\mathcal{S}$ stalls the main execution and proceeds to rewind $\mathcal{A}$. Specifically, $\mathcal{S}$ rewinds $\mathcal{A}$ to the third message and supplies a different second message for $\pi_{\mathrm{DL}}^{\mathrm{ZK}}$ by sampling uniformly random new challenge e'_{Sen}. If $e_{\mathrm{Sen}} = e'_{\mathrm{Sen}}$, i.e., the challenge is identical, then $\mathcal{S}$ aborts. Otherwise, it feeds the challenge to $\mathcal{A}$ as part of the second message. Finally, $\mathcal{S}$ runs the extractor for the WI-PoK $\pi_{\mathrm{DL}}^{\mathrm{WI}}$ and extracts the bit b and the discrete logarithm of PK_{1-b}.

   Specifically, let γ be such that the simulator extracts $\widetilde{m}$ with respect to PK_γ. Then $\mathcal{S}$ fixes the bit $b = 1 - \gamma$.
3. $\mathcal{S}$ submits b to $\mathcal{F}_{\mathrm{OT}}$, receiving back s_b.
4. $\mathcal{S}$ rewinds $\mathcal{A}$ to the third message and computes it based on s_b and random s_{1-b}.
5. $\mathcal{S}$ halts, outputting whatever $\mathcal{A}$ does.

Note first that the simulator runs in polynomial-time and that the probability it aborts is negligible. Moreover, we prove that the simulated and real views are computationally indistinguishable via a reduction to the security of the El Gamal PKE. Namely, we prove the following claim,

Claim 4.2. *The following two distribution ensembles are computationally indistinguishable,*

$$\left\{\mathbf{View}_{\pi_{\mathrm{OT}},\mathcal{A}(z)}(n,(s_0,s_1),b)\right\}_{n\in\mathbb{N},s_0,s_1,b,z\in\{0,1\}^*} \stackrel{c}{\approx} \left\{\mathbf{View}_{\mathcal{F}_{\mathrm{OT}},\mathcal{S}(z)}(n,(s_0,s_1),b)\right\}_{n\in\mathbb{N},s_0,s_1,b,z\in\{0,1\}^*}.$$

Proof: Assume by contradiction that these two views are distinguishable by a PPT distinguisher D. We construct an adversary $\mathcal{A}'$ that breaks the security of the El Gamal PKE as follows. Recall that $\mathcal{A}'$ externally communicates with a challenger that provides to it a public key $\mathrm{PK} = \langle g, h\rangle$ and a challenge ciphertext. Upon receiving PK and (s_0, s_1) as the auxiliary input, $\mathcal{A}'$ picks a random bit β' and sets $h_{\beta'} = g^x$ for some random $x \leftarrow \mathbb{Z}_p$. In addition, $\mathcal{A}'$ sets $h_{1-\beta'} = h/h_{\beta'}$. $\mathcal{A}'$ invokes $\mathcal{A}$ internally and forwards it the first message of the protocol g, h_0, h_1. Upon receiving $\mathcal{A}$'s second message, $\mathcal{A}'$ aborts if $\beta' \neq \beta$. Else, $\mathcal{A}'$ completes the execution using arbitrary (s_0, s_1). Upon completing the execution successfully, $\mathcal{A}'$ extracts b and the discrete logarithm of PK_{1-b} exactly as done in the simulation. Finally, $\mathcal{A}'$ submits to its challenger the two messages $s_{1-b}^{\widetilde{m}^{-1}}$ and t for $t \leftarrow \mathbb{Z}_p$, receiving back a challenge ciphertext $c = \langle c'_0, c'_1\rangle$ that encrypts one of these plaintexts at random. $\mathcal{A}'$ computes $\langle (c'_0)^{\widetilde{m}}, (c'_1)^{\widetilde{m}}\rangle$ (and rerandomizes the ciphertext by multiplying the outcome with a random encryption of zero), and plugs the outcome instead of the ciphertext that encrypts s_{1-b} and halts. Finally, $\mathcal{A}'$ invokes D on the joint distribution of (s_0, s_1) and the adversary's output and outputs whatever D does.

We now consider two cases:

1. In the first case the challenge c' is an encryption of $s_{1-b}^{\widetilde{m}^{-1}}$. We claim that in this case the adversary's view is distributed as in the real execution. This is because the challenge ciphertext $\langle (c_0')^{\widetilde{m}}, (c_1')^{\widetilde{m}} \rangle$ corresponds to a random ciphertext that encrypts the plaintext s_{1-b} relative to PK_{1-b}.

2. On the other hand, in case the challenge c' is an encryption of a random element t, then the adversary's view is distributed as in the simulation, as the simulator does not know s_{1-b} and hence uses a random input instead of the real value.

   In both cases, the first message of the reduction is identically distributed to the first message in the corresponding execution. Moreover, the distribution of the first message for $\beta' = 0$ is identical to the distribution for the case that $\beta' = 1$.

More formally, assume that

$$\left| \Pr[D(\mathbf{View}_{\pi_{\mathrm{OT}},\mathcal{A}(z)}(n,(s_0,s_1),b)) = 1] - \Pr[D(\mathbf{View}_{\mathcal{F}_{\mathrm{OT}},\mathcal{S}(z)}(n,(s_0,s_1),b)) = 1] \right| \geq \varepsilon(n).$$

Then, it holds that

$$\Pr[\textsc{Adv}_{\Pi,\mathcal{A}'}(n) = 1] \geq \frac{1}{2} + \frac{\varepsilon(n)}{2}$$

condition on the event for which $\beta' = \beta$. This is proven as follows,

$$\begin{aligned}
\Pr[\textsc{Adv}_{\Pi,\mathcal{A}'}(n) = 1] &= \frac{1}{2}\Big(\Pr[\textsc{Adv}_{\Pi,\mathcal{A}'}(n) = 1 | b = 0] + \Pr[\textsc{Adv}_{\Pi,\mathcal{A}'}(n) = 1 | b = 1] \Big) \\
&= \frac{1}{2}\Big(\Pr[D(\mathbf{View}_{\pi_{\mathrm{OT}},\mathcal{A}(z)}(n,(s_0,s_1),b)) = 0] + \Pr[D(\mathbf{View}_{\mathcal{F}_{\mathrm{OT}},\mathcal{S}(z)}(n,(s_0,s_1),b)) = 1] \Big) \\
&= \frac{1}{2}\Big(1 - \Pr[D(\mathbf{View}_{\pi_{\mathrm{OT}},\mathcal{A}(z)}(n,(s_0,s_1),b)) = 1] + \Pr[D(\mathbf{View}_{\mathcal{F}_{\mathrm{OT}},\mathcal{S}(z)}(n,(s_0,s_1),b)) = 1] \Big) \\
&= \frac{1}{2} + \frac{1}{2}\Big| \Pr[D(\mathbf{View}_{\mathcal{F}_{\mathrm{OT}},\mathcal{S}(z)}(n,(s_0,s_1),b)) = 1] - \Pr[D(\mathbf{View}_{\pi_{\mathrm{OT}},\mathcal{A}(z)}(n,(s_0,s_1),b)) = 1] \Big| \\
&\geq \frac{1}{2} + \frac{\varepsilon(n)}{2}.
\end{aligned}$$

□ □

4.3 Obtaining Secure Two-Party Computation

First we observe that we can repeat our OT protocol in parallel guaranteeing the same security. Then, obtaining general secure two-party computation is carried out by embedding the two-round protocol of [IKO+11] within our second/third messages of our OT protocol. Namely, upon receiving the third message, the receiver computes the OT outcome and uses this to compute the outcome of the [IKO+11] protocol. It then sends this outcome together with the OT last message to the sender, that concludes the outcome of the general computation upon verifying the OT message correctly. We remark that we obtain a secure two-party protocol with the same security guarantees, namely, security against malicious non-aborting senders and malicious non-aborting receivers.

In more details, in order to achieve simulation when the sender is corrupted, we observe that, upon extracting the trapdoors, it is possible to set up the OT part in the second message from the receiver in such a way that the sender's inputs to the OT can be extracted with perfect simulation. Then relying on the simulation of the [IKO+11] simulator in the $\mathcal{F}_{\mathrm{OT}}$-hybrid we carry out the rest of the simulation. To achieve simulation when the receiver is corrupted, we consider a simulator that honestly generates the sender's messages with arbitrary inputs for the functionality being computed and then extracts the receiver's inputs to the OT by rewinding the WI-PoK. Using the inputs to the OT, we then rely on the simulation of the malicious receiver in the $\mathcal{F}_{\mathrm{OT}}$-hyrbid in the [IKO$^+$11] protocol. Thus, we obtain the following theorem:

Theorem 4.2. *Assuming the Decisional Diffie-Hellman problem is hard, there exists a four-round secure computation protocol for any functionality that is secure in the presence non-aborting adversaries.*

5 4-Round Two-Party Computation with $1/\mathrm{p}$ Sender Security

In this section we extend our protocol from Section 4 and demonstrate how to achieve $1/p$-simulation with respect to corrupted aborting senders. Our protocol is inspired by the recent result of Ostrovsky, Richelson and Scafuro [ORS15] and relies on the following additional building blocks: let (1) Commit be a statistically binding commitment scheme, (2) let $(\mathsf{Share}, \mathsf{Rec})$ be a $(M+1)$-out-of-$2M$ Shamir secret-sharing scheme over $\mathbb{Z}_q$, together with a linear map $\phi : \mathbb{Z}_q^{2M} \to \mathbb{Z}_q^{M-1}$ such that $\phi(v) = 0$ iff v is a valid sharing of some secret. We further note that the WI-PoK $\pi_{\mathrm{DL}}^{\mathrm{WI}}$ that is given by Rec in Protocol 3, is extended here to handle the parallel case. Namely, the receiver proves the validity of one of the public keys it generates within each pair, in parallel. On a high-level, we modify Protocol 3 as follows.

- We repeat Protocol 3 in parallel $3M$ times to obtain $3M$ oblivious transfer parallel executions. We divide this set of executions into two sets of M and $2M$ executions.

- The sender chooses first two random inputs $x_0, x_1 \in \mathbb{Z}_q$ and secret shares them using the Shamir secret-sharing scheme to obtain shares $[x_0]$ and $[x_1]$. Next, for $b \in \{0,1\}$ it picks M pairs of vectors that add up to $[x_b]$. It is instructive to view them as matrices $A_0, B_0, A_1, B_1 \in \mathbb{Z}_q^{M\times 2M}$ where for every row $i \in [M]$ and $b \in \{0,1\}$, it holds that $A_b[i,\cdot] \oplus B_b[i,\cdot] = [x_b]$. Next, the sender commits to each entry of each matrix separately in the third message of the protocol. To check the validity of the shares the sender additionally sends matrices Z_0, Z_1 in the clear, such that the row $Z_b[i,\cdot]$ is set to $\phi(A_b[i,\cdot])$, along with the third message of the protocol where it commits to the entries of A_0, A_1, B_0 and B_1. Finally, it sends $C_0 = x_0 + s_0$ and $C_1 = x_1 + s_1$.

- In the first set of M OT executions, the sender's input to the ith execution is the decommitment information of the entire ith row

$$((A_0[i,\cdot], A_1[i,\cdot]), (B_0[i,\cdot], B_1[i,\cdot]))\,,$$

  whereas the receiver sets its input to these executions as $c_1, \ldots, c_M$ at random. Upon receiving its output for the OT, the receiver proceeds as follows: If $c_i = 0$, then the receiver checks whether $\phi(A_b[i,\cdot]) = [z_i^b]$, and if $c_i = 1$ it checks whether $\phi(B_b[i,\cdot]) + Z_b[i,\cdot] = 0$. This is referred to as the *shares validity check.*

- In the second set of $2M$ OT executions, the sender's input to the jth OT execution is the decommitment information of the entire jth column

$$((A_0[\cdot,j], B_0[\cdot,j]), (A_1[\cdot,j], B_1[\cdot,j]))\,.$$

  Looking ahead, if the receiver's input is b, then upon receiving its output for a particular column j it checks that for all i, $A_b[i,j] \oplus B_b[i,j]$ agree on the same value. We refer to this as the *shares consistency check.*

- In the second set of OTs, the receiver sets its input as follows. It selects a random subset $T_{1-b} \subseteq [2M]$ of size $M/2$ and defines $T_b = [2M]/T_{1-b}$. Then, for every $j \in [2M]$, Rec sets $b_j = \beta$ if $j \in T_\beta$. The b_j values serve as the inputs to the OT for the next $2M$ executions.

- Finally, the receiver first checks if all the rows obtained from the first set of OT executions pass the shares validity check. Next, it checks if all the columns in T_{1-b} and a random subset of size $M/2$ from T_b pass the shares consistency check. If so, it finds $M+1$ columns in T_b that pass the shares consistency check, extracts the share that corresponds to each such column and then uses these $M+1$ shares to reconstruct x_b. Finally, the receiver uses x_b and C_b to compute s_b.
- Additionally, we modify the WI-PoK to a proof for a statement that captures all parallel executions simultaneously, i.e. the statements of all OT executions are combined using the logical AND.

The security guarantees of this protocol are $1/p$-simulation against malicious senders and full security against non-aborting malicious receivers. We remark that the receiver's simulation essentially follows a similar approach as in the simulation of Protocol 3, where it rewinds the WI-PoK protocol in order to extract the receiver's inputs to all the parallel OT executions and then setting the input that the receiver cannot obtain to a random string (one at a time), concluding that there will not be enough information for any receiver to extract s_{1-b}. On the other hand, the sender simulation needs to achieve $1/p$-simulation. The high-level idea is to apply techniques from the simulation in [ORS15], given that the simulator extracts sufficiently enough shares of the sender's inputs to the parallel OTs. The core of our argument and the main technical part of this protocol is to show that if an adversarial sender does not abort before sending the third message too often (i.e. $< 1 - \frac{1}{p}$) then the simulator can extract the trapdoor by rewinding sufficiently many times. Namely, in this case, we show that the simulator can extract the discrete logarithm of both h_0 and h_1 with respect to g in at least $1 - \frac{1}{3p}$ fraction of the OT executions. Then we can show that the simulator succeeds in extracting the sender's inputs s_0, s_1 with very high-probability.

5.1 Our OT Protocol

We are now ready to describe our protocol for secure computation. We construct a four-round OT protocol with the stronger guarantee of $\frac{1}{p}$-security in the presence of (possibly aborting) malicious senders.

Protocol 4 (Protocol π_{OT})**.**

Public parameters: *The description of a group $\mathbb{G}$ of prime order p.*

Inputs: *The sender* Sen *holds s_0, s_1 and the receiver* Rec *holds a bit b.*

The protocol:

1. **Sen $\rightarrow$ Rec :**
   (a) *Let $N = 3M$. Then, for $i \in [N]$,* Sen *picks random generator $g_i \leftarrow \mathbb{G}$ and computes $h_{i,0} = g^{r_{i,0}}$ and $h_{i,1} = g_i^{r_{i,1}}$ where $r_{i,0}, r_{i,1} \leftarrow \mathbb{Z}_p$.*
   (b) Sen *sends the N tuples $\{g_i, h_{i,0}, h_{i,1}\}_{i\in[N]}$ to* Rec.
2. **Rec $\rightarrow$ Sen :**
   (a) Rec *samples uniformly at random $c_1, \ldots, c_M \leftarrow \{0,1\}$. The c_i values serve as the input to the first M OT executions.*
   (b) Rec *selects a random subset $T_{1-b} \subseteq [2M]$ of size $M/2$. Define $T_b = [2M]/T_{1-b}$. For every $j \in [2M]$,* Rec *sets $b_j = \alpha$ if $j \in T_\alpha$. The b_j values serve as the inputs to the OT for the next $2M$ executions.*
   (c) *According to its input for the $3M$ OT executions,* Rec *generates $N = 3M$ pairs of El Gamal PKE's as follows:*
      - *For every $i \in [M]$, $\mathrm{PK}_{i,c_i} = g_i^{m_i}$ and $\mathrm{PK}_{i,1-c_i} = (h_{i,0}h_{i,1})^{\widetilde{m}_i}$ where $m_i, \widetilde{m}_i \leftarrow \mathbb{Z}_p$.* Rec *sets* $\mathrm{SK}_i = m_i$.
      - *For every $j \in [2M]$, $\mathrm{PK}_{M+j,b_j} = g_{M+j}^{m_{M+j}}$ and $\mathrm{PK}_{M+j,1-b_j} = (h_{M+j,0}h_{M+j,1})^{\widetilde{m}_{M+j}}$ where $m_{M+j}, \widetilde{m}_{M+j} \leftarrow \mathbb{Z}_p$.* Rec *sets* $\mathrm{SK}_{M+j} = m_{M+j}$.

(d) Rec *sends* $\{\text{PK}_{i,0}, \text{PK}_{i,1}\}_{i\in[N]}$ *to* Sen.

(e) Rec *sends the first message of the WI-PoK for proving the knowledge for every* $i \in [N]$ *of the discrete logarithms of either* PK_0^i *or* PK_1^i *with respect to* $(h_{i,0}h_{i,1})$.

(f) Rec *sends a challenge string* $\beta = (\beta_1, \ldots, \beta_N)$.

(g) Rec *sends the first message for the statistically-binding commitment scheme* com.

3. **Sen → Rec :**

(a) Sen *picks two random strings* $x_0, x_1 \leftarrow \mathbb{Z}_q$ *and secret shares them using the Shamir's secret-sharing scheme. In particular,* Sen *computes* $[x_b] = (x_b^1, \ldots, x_b^{2M}) \leftarrow \mathsf{Share}(x_b)$ *for* $b \in \{0,1\}$. Sen *commits to the shares* $[x_0], [x_1]$ *as follows. It picks random matrices* $A_0, B_0 \leftarrow \mathbb{Z}_q^{M\times 2M}$ *and* $A_1, B_1 \leftarrow \mathbb{Z}_q^{M\times 2M}$ *such that* $\forall i \in [M]$*:*

$$A_0[i,\cdot] + B_0[i,\cdot] = [x_0], \;\; A_1[i,\cdot] + B_1[i,\cdot] = [x_1].$$

Sen *computes two matrices* $Z_0, Z_1 \in \mathbb{Z}_q^{M\times M-1}$ *and sends them in the clear such that:*

$$Z_0[i,\cdot] = \phi(A_0[i,\cdot]), Z_1[i,\cdot] = \phi(A_1[i,\cdot]).$$

(b) Sen *sends the committed matrices* $(\mathsf{com}_{A_0}, \mathsf{com}_{B_0}, \mathsf{com}_{A_1}, \mathsf{com}_{B_1})$ *to* Rec *where each element of each matrix is individually committed using* com.

(c) *For* $i \in [M]$*,* Sen *computes ciphertexts* $c_{i,0}, c_{i,1}$ *where* $c_{i,0}$ *is an encryption of the decommitment of the rows* $A_0[i,\cdot]$ *and* $A_1[i,\cdot]$ *under public key* $\text{PK}_{i,0}$ *and* $c_{i,1}$ *is an encryption of the decommitment of the rows* $B_0[i,\cdot]$ *and* $B_1[i,\cdot]$ *under public key* $\text{PK}_{i,1}$. Sen *sends* $\{c_{i,0}, c_{i,1}\}_{i\in[M]}$ *to* Rec.

(d) *For* $j \in [2M]$*,* Sen *computes ciphertexts* $\tilde{c}_{j,0}, \tilde{c}_{j,1}$*, where* $\tilde{c}_{j,b}$ *is an encryption of the decommitment of the columns* $A_b[\cdot,j], B_b[\cdot,j]$ *under public key* $\text{PK}_{M+j,b}$. Sen *sends* $\{\tilde{c}_{j,0}, \tilde{c}_{j,1}\}_{j\in[2M]}$ *to* Rec.

(e) Sen *sends the second message* e_{Sen} *for the WI-PoK protocol* $\pi_{\text{DL}}^{\text{WI}}$ *given by the receiver (recall that this message is a random challenge).*

(f) Sen *sends* $r_{\beta_i} = \log_{g_i}(h_{i,\beta})$ *for all* $i \in [N]$.

(g) Sen *sends* $C_0 = s_0 \oplus x_0$ *and* $C_1 = s_1 \oplus x_1$ *to* Rec.

4. **Rec → Sen :**

(a) **Decryption Phase:** *Upon receiving the all the sender's ciphertexts the receiver decrypts them to obtain the OT outputs. These include decommitments to* $A_0[i,\cdot], A_1[i,\cdot]$ *for every* $i \in [M]$ *when* $c_i = 0$ *and decommitments to* $B_0[i,\cdot], B_1[i,\cdot]$ *when* $c_i = 1$*. They also include columns* $A_{b_j}[\cdot,j], B_{b_j}[\cdot,j]$ *for every* $j \in [2M]$.

(b) **Shares Validity Check Phase:** *For* $i = 1, \ldots, M$*, if* $c_i = 0$ *check that* $Z_0[i,\cdot] = \phi(A_0[i,\cdot])$ *and* $Z_1[i,\cdot] = \phi(A_1[i,\cdot])$*. Otherwise, if* $c_i = 1$ *check that* $\phi(B_0[i,\cdot]) + Z_0[i,\cdot] = 0$ *and* $\phi(B_1[i,\cdot]) + Z_1[i,\cdot] = 0$*. If the tokens do not abort and all the checks pass, the receiver proceeds to the next phase.*

(c) **Shares Consistency Check Phase:** *For each* $b \in \{0,1\}$*,* Rec *randomly chooses a set* T_b *for which* $b_j = b$ *at* $M/2$ *coordinates. For each* $j \in T_b$*,* Rec *checks that there exists a unique* x_b^i *such that* $A_b[i,j] + B_b[i,j] = x_b^j$ *for all* $i \in [M]$*. If so,* x_b^j *is marked as consistent. If all shares obtained in this phase are consistent,* Rec *proceeds to the reconstruction phase. Else it aborts.*

(d) **Reconstruction Phase:** *For* $j \in [2M]/T_{1-b}$*, if there exists a unique* x_b^j *such that* $A_b[i,j]+B_b[i,j] = x_b^j$*,* Rec *marks share* j *as a consistent column. If* R *obtains less than* $M+1$ *consistent columns, it aborts. Otherwise, let* $x_b^{j_1}, \ldots, x_b^{j_{M+1}}$ *be any set of* $M+1$ *shares obtained from consistent columns.* Rec *computes* $x_b \leftarrow \mathsf{Reconstruct}(x_b^{j_1}, \ldots, x_b^{j_{M+1}})$ *and outputs* $s_b = C_b \oplus x_b$.

(e) Rec *sends the last message for the WI-PoK protocol* $\pi_{\text{DL}}^{\text{WI}}$.

Theorem 5.1. *Assume that the Decisional Diffie-Hellman assumption holds in* $\mathbb{G}$ *and that* $\pi_{\text{DL}}^{\text{WI}}$ *is as above. Then, Protocol 4 securely realizes* $\mathcal{F}_{\text{OT}}$ *in the presence of* non-aborting *static, malicious receivers and aborting static malicious senders (in the sense of Definition A.4 with* $\frac{1}{p}$ *security) .*

Proof: We consider each corruption case separately.

Sen **is corrupted.** Recall that when the sender is corrupted we need to prove $1/p$-indistinguishability. More precisely, we need to define a simulator that produces a view of the malicious sender $\mathcal{A}$ while extracting both s_0 and s_1, where the view and the value learned by the honest receiver is $1/p$-indistinguishable from the sender's real view and the receiver's output in a real execution. More precisely, for any probabilistic polynomial-time adversary $\mathcal{A}$ controlling Sen we define a simulator $\mathcal{S}$ that proceeds as follows:

1. $\mathcal{S}$ invokes $\mathcal{A}$ on its input and randomness of appropriate length.
2. Upon receiving from $\mathcal{A}$ the first message, $\mathcal{S}$ computes the second message honestly with input $b = 0$. If $\mathcal{A}$ aborts before sending the third message then $\mathcal{S}$ outputs the view of $\mathcal{A}$ and halts.
3. Otherwise, upon receiving $\mathcal{A}$'s third message, $\mathcal{S}$ records the set $\{r_{\beta_i}\}_{i\in[N]}$. Next, it stalls the main execution and proceeds to rewind $\mathcal{A}$. Specifically, $\mathcal{S}$ rewinds $\mathcal{A}$ to the second message and proceeds as follows:
   - For every $i \in [N]$ and $\gamma \in \{0,1\}$, $\mathcal{S}$ rewinds $\mathcal{A}$ for $T = N^4p$ attempts, where in each such attempt $\mathcal{S}$ supplies a uniformly random second message according to the receiver's strategy with input $b = 0$, where $\beta_i = \gamma$. In each rewinding, $\mathcal{S}$ collects the correct discrete logarithms within $\mathcal{A}$'s reply.
4. If upon concluding the rewinding phase $\mathcal{S}$ does not obtain the discrete logarithm of both $h_{i,0}$ and $h_{i,1}$ for at least $1 - 1/3p$ fraction of $i \in [N]$, then it halts outputting fail.
5. Otherwise, let $I \subseteq \{1, \ldots, M\}$ and $J \subseteq \{M+1, \ldots, 3M\}$ be the sets of indices for which it extracts both the discrete logarithms. We remark that $\mathcal{S}$ does not try to extract the sender's inputs in the first M executions, namely, for indices in I. Next, $\mathcal{S}$ rewinds $\mathcal{A}$ back to the second message and for every $j \in J$ generates public keys so that it can extract both the sender's inputs. For all other executions, it follows the honest receiver's strategy corresponding to the input $b = 0$. $\mathcal{S}$ completes the execution by using the witness corresponding to $b = 0$ for the receiver in the WI-PoK.[3] Upon completion, it performs the share consistency check and the share validity check as the honest receiver would and if either of them fail, then the simulator halts outputting the view of $\mathcal{A}$.
6. Otherwise, it decrypts all ciphertexts for which it knows the corresponding secret keys. For each $b \in \{0,1\}$ and $j \in J$, if there exists a unique x_b^j such that $A_b[i,j] + A_b[i,j] = x_b^j$, $\mathcal{S}$ marks column $j - M$ as consistent. If it obtains at least $M+1$ shares for x_b from consistent columns it reconstructs x_b, and then obtains s_b from x_b and C_b. If not, it sets $s_b = \perp$.
7. Finally, $\mathcal{S}$ forwards (s_0, s_1) to $\mathcal{F}_{\mathrm{OT}}$ and halts, outputting whatever $\mathcal{A}$ does.

□

Clearly, $\mathcal{S}$ runs in strict polynomial-time. We next prove the correctness of simulation. On a high-level, the second message in all the rewinding attempts is generated identically to the second message of the real execution, and is independent of the bit b that the receiver enters. This follows by repeating Protocol 3 in parallel, for which the indistinguishability argument is similar. Let $s \in \omega(\log n)$. Two cases arise:

1. *The abort probability of the sender is higher than* $1 - \frac{1}{Np}$. In this case, $1/p$-indistinguishability is achieved directly as the simulation outputs views on which the sender aborts with at least the same probability as such views occur in the real view. Now, since this accounts for a probability mass of at least $1 - \frac{1}{Np} > 1 - \frac{1}{p}$, $1/p$-indistinguishability follows.

[3]This is possible because for indices outside J it has the correct witness, and for indices in J it has witnesses corresponding to both inputs of the receiver.

2. *The abort probability of the sender is at most* $1-\frac{1}{Np}$. In this case by setting $M > sp$, for s being some superlogarithmic function in n, we argue that except with negligible probability (roughly, $2^{-O(s)}$), the simulator will be able to obtain the discrete logarithms of both $h_{i,0}$ and $h_{i,1}$, i.e., the trapdoors, for at least $1-\frac{1}{3p}$ fraction of the indices $i \in [3M]$ via rewinding. This is formally proven in Claim 5.1. Just as in the previous protocol, we have that for every index in $\{M+1,\dots,3M\}$ that the simulator obtains a trapdoor, it will be able to extract both of the sender's inputs. Specifically, as $M > sp$, we can conclude that the simulator fails to obtain the trapdoor of at most $\frac{1}{3p} \times 3M = s$ indices. This means that among the indices in $\{M+1,\dots,3M\}$ it obtains trapdoors for at least $2M-s$ indices.

   Next, from the shares consistency check we can conclude that with very high probability all but s columns contain shares that are consistent. From the shares validity check we can conclude that with very high probability there is a single row i_b corresponding to each $b \in \{0,1\}$ such that $A_b[i,\cdot] + B_b[i,\cdot]$ contains valid shares of some secret. Combining these checks, we can conclude that there are at least $2M-s$ columns that are consistent, i.e., the shared value in each row is the same and therefore must equal $A_b[i_b,\cdot] + B_b[i_b,\cdot]$. Furthermore, from the statistically binding property of the commitment scheme Commit we can conclude that for any one of these consistent columns, there can be only one value for the shares that can be extracted by both the receiver and the simulator.

   In this case, we can now conclude that, using the trapdoors, the simulator obtains at least $2M-s-s$ shares for both inputs among the consistent columns. Since $M > sp$ we have that $2M-2s > M+1$ (for $p > 2$) and from $M+1$ valid shares it can extract s_b for each $b \in \{0,1\}$.

Claim 5.1. *We say that $i \in [N]$ is* extractable*, if $\mathcal{S}$ manages to extract the discrete logarithms of both $h_{i,0}$ and $h_{i,1}$ with respect to g_i. If the adversary $\mathcal{A}$ does not abort before sending the third message with probability at least $\frac{1}{Np}$, then except with negligible probability, at least $1-\frac{1}{3p}$ fraction of the indices are* extractable.

Proof: On a high-level we follow a Yao-type amplification argument [Yao82]. First, we observe that the distribution of the second message fed to $\mathcal{A}$ in any rewinding attempt is perfectly distributed to the real distribution. Next, suppose that $\mathcal{A}$ does not abort with probability at least $\frac{1}{N^3p}$ both when its view is conditioned on $\beta_i = 0$ and when it is conditioned on $\beta_i = 1$, for some index i. Then we show that i is extractable except with negligible probability. This is because for every i and every value of β_i the simulator makes N^4p rewinding attempts, thus the probability that it fails to find a successful execution where the adversary $\mathcal{A}$ responds is at most $\left(1-\frac{1}{N^3p}\right)^{N^4p} = O(e^{-s})$. Therefore, it suffices to show that this condition holds for more than $1-\frac{1}{3p}$ indices i for which $\mathcal{A}$ does not abort with probability at least $\frac{1}{N^3p}$ both when its view is conditioned on $\beta_i = 0$ and when it is conditioned on $\beta_i = 1$. This is because using the preceding argument, we can conclude that at least $1-\frac{1}{3p}$ fraction of indices are extractable and the proof of the claim follows.

Suppose for contradiction that there are more than $\frac{1}{3p}$ fraction of indices for which the condition does not hold. This means that for a set of $\frac{1}{3p}N = s$ indices, denoted by S, and values $\{\gamma_j\}_{j\in S}$ such that for every $j \in S$, when conditioned on $\beta_j = \gamma_j$, the probability that the adversary aborts is greater than $1-\frac{1}{N^3p}$.

We now estimate the overall probability that $\mathcal{A}$ does not abort.

$$\begin{aligned}
\Pr[\mathcal{A} \text{ does not abort}] &= \Pr[\mathcal{A} \text{ does not abort} \mid \exists\, j \in S \text{ s.t. } \beta_j = \gamma_j] \Pr[\,\exists\, j \in S \text{ s.t. } \beta_j = \gamma_j] \\
&\qquad + \Pr[\mathcal{A} \text{ does not abort} \mid \forall\, j \in S, \beta_j \neq \gamma_j] \Pr[\,\forall\, j \in S, \beta_i \neq \gamma_i] \\
&\leq \frac{N}{N^3 p} \times \Pr[\,\exists\, j \in S \text{ s.t. } \beta_j = \gamma_j] + 1 \times \Pr[\,\forall\, j \in S, \beta_j \neq \gamma_j] \\
&\leq \frac{N}{N^3 p} + \frac{1}{2^s} \\
&\leq \frac{1}{2N^2 p}.
\end{aligned}$$

This is a contradiction since we assumed that $\mathcal{A}$ does not abort with probability at least $\frac{1}{Np}$. □

Finally, to argue $1/p$-indistinguishability, we consider two cases:

Case: non-aborting probability of $\mathcal{A}$ is greater than $\frac{1}{pN}$. First, we observe that the sender's view in the simulation is statistically close to the real view. This follows using an argument analogous to our previous protocol as the public-keys in the second message (even those generated by the simulation using the trapdoors) and the perfect WI-PoK perfectly hide the receiver's input. It therefore suffices to argue that the receiver's messages can be generated while extracting the sender's input. Using the preceding argument, we have that the simulation will always succeed in extracting the trapdoors of at least $1 - \frac{1}{3p}$ fraction of the parallel OT executions. Since $M > sp$, we can conclude that the simulator fails to obtain the trapdoor of at most $\frac{1}{3p} \times 3M = s$ indices. This means that among the indices in $\{M+1, \ldots, 3M\}$ it obtains trapdoors for at least $2M - s$ indices. Recall that, after extraction, the simulator rewinds the sender to the second message and generates the receiver's message by setting up the public-keys as follows: for every index in J the simulator uses the trapdoor and sets the public-keys so that it can extract both of the sender's inputs. For the rest of the indices, it simply sets the receiver's input bit to 0.

Next, from the shares consistency check we can conclude that with very high probability all but s columns contain shares that are consistent. Moreover, the share validity check makes the receiver check if $Z_b[i,\cdot] = \phi(A_b[i,\cdot])$ holds or $Z_b[i,\cdot] + \phi(B_b[i,\cdot]) = 0$ holds. If for a row both conditions hold, then we have the $\phi(A_b[i,\cdot]) + \phi(B_b[i,\cdot]) = 0$ and $A_b[i,\cdot] + B_b[i,\cdot]$ must contain a valid vector of shares. Now even if one of these two conditions fail to hold for more than s rows, the sender will be caught with probability $1 - 2^{-s}$. Therefore, there are at least $M - s$ rows for which $\phi(A_b[i,\cdot]) + \phi(B_b[i,\cdot]) = 0$. For our argument, it suffices to have just one row i_b corresponding to each $b \in \{0,1\}$ such that $A_b[i,\cdot] + B_b[i,\cdot]$ contains valid shares of some secret. Combining these checks, we can conclude that there are at least $2M - s$ columns that are consistent, i.e., the shared value in each row is the same and must equal $A_b[i_b,\cdot] + B_b[i_b,\cdot]$. Furthermore, from the statistically binding property of the commitment scheme Commit we can conclude that for any one of these consistent columns, there can be only one value for the shares that can be extracted by both the receiver and the simulator.

In this case, we can now conclude that, using the trapdoors, the simulator obtains at least $2M - s - s$ shares for both inputs among the consistent columns. Since $M > sp$ we have that $2M - 2s > M + 1$ (for $p > 2$) and from $M + 1$ valid shares it can extract s_b for each $b \in \{0,1\}$. Furthermore, the sender's input extracted by the honest receiver while holding the input b and the input extracted by the simulator have to be the same as both of them have to correspond to the shares in the row i_b.

Case: non-aborting probability of $\mathcal{A}$ is at most $\frac{1}{pN}$. From the first step of the simulation we know that all views on which $\mathcal{A}$ aborts are simulated at least with the same probability as in the real view. Now, if the non-aborting probability is smaller than $\frac{1}{pN}$ then the probability mass of aborting views is at least $1 - \frac{1}{pN} > 1 - \frac{1}{p}$ and we achieve $1/p$-indistinguishability.

Thus, we have the following claim.

Claim 5.2. *The following two distribution ensembles are identical,*

$$\left\{\mathbf{View}_{\pi_{\mathrm{OT}},\mathcal{A}(z)}(n,(s_0,s_1),b)\right\}_{n\in\mathbb{N},s_0,s_1,b,z\in\{0,1\}^*} \overset{1/\mathrm{p}}{\approx} \left\{\mathbf{View}_{\mathcal{F}_{\mathrm{OT}},\mathcal{S}(z)}(n,(s_0,s_1),b)\right\}_{n\in\mathbb{N},s_0,s_1,b,z\in\{0,1\}^*}.$$

Proof: The proof follows essentially using the same ideas from the previous protocol. □

Rec **is corrupted.** In this case we need to prove that any **non aborting** corrupted receiver cannot learn anything about the sender's other input s_{1-b} while extracting b. More precisely, for any probabilistic polynomial-time adversary $\mathcal{A}$ controlling Rec we define a simulator $\mathcal{S}$ that proceeds as follows:

1. $\mathcal{S}$ invokes $\mathcal{A}$ on its input and randomness of the appropriate length.

2. $\mathcal{S}$ plays the role of the honest sender with arbitrary inputs (s'_0, s'_1). Upon completing the execution successfully, $\mathcal{S}$ stalls the main execution and proceeds to rewind $\mathcal{A}$. Specifically, $\mathcal{S}$ rewinds $\mathcal{A}$ to the third message and supplies a different second message for $\pi^{\mathrm{ZK}}_{\mathrm{DL}}$ by sampling uniformly random new challenge e'_{Sen}. If $e_{\mathrm{Sen}} = e'_{\mathrm{Sen}}$, i.e., the challenge is identical, then $\mathcal{S}$ aborts. Otherwise, it feeds the challenge to $\mathcal{A}$ as part of the second message. Finally, $\mathcal{S}$ runs the extractor for the WI-PoK $\pi^{\mathrm{WI}}_{\mathrm{DL}}$ and extracts the inputs to all the OT executions along with the discrete logarithm of the corresponding key.

3. Among the executions $M+1,\ldots,3M$, $\mathcal{S}$ finds that bit b that occurs at least $M+1$ times and submits b to $\mathcal{F}_{\mathrm{OT}}$, receiving back s_b. Recall that since the receiver is a non-aborting adversary, it completes the protocol without allowing the honest sender to abort. In other words, it convinces the sender in the WI-PoK with probability 1. Therefore, since a witness will be extracted from the proof-of-knowledge, the input of the receiver in the parallel OTs are well defined. Specifically, $\mathcal{S}$ extracts the adversary's inputs to these OT executions as in the simulation for Protocol 3.

4. $\mathcal{S}$ rewinds $\mathcal{A}$ to the third message and computes it based on s_b and random s_{1-b}.

5. $\mathcal{S}$ halts, outputting whatever $\mathcal{A}$ does.

Note first that the simulator runs in polynomial-time and that the probability it aborts is negligible. Moreover, we prove that the simulated and real views are computationally indistinguishable via a reduction to the security of the El Gamal PKE. We provide a brief proof sketch below:

- From the proof of Protocol 3, using the privacy argument of the El Gamal PKE, we know that if for a particular OT execution the sender (played by the simulator) extracted the receiver's input as γ, then the sender's input that corresponds to the bit $1-\gamma$ can be replaced by a random value. We consider a sequence of hybrids where we replace at least one input in each of the $3M$ executions with a random input. More formally, for every $j \in \{1,\ldots,M\}$, depending on what value is extracted for each of the c_i, and every $b \in \{0,1\}$, we replace the sender's input containing the decommitment of $A_b[i,\cdot]$ with random or that containing the decommitment of $B_b[i,\cdot]$ with random. Next, for $j \in \{M+1,\ldots,2M\}$ depending on the value extracted as b_i for the receiver, we replace the input containing the decommitment of $(A_0[\cdot,j-M], B_0[\cdot,j-M])$ random or the other input containing the decommitment of $(A_1[\cdot,j-M], B_1[\cdot,j-M])$ random. Next, in another sequence of hybrids, for every value that is set to random we also replace the corresponding commitment to a random value.

- Next, we argue that at least M shares of x_{1-b} (out of the $2M$ shares) are hidden, where b is the adversary's input as extracted in the simulation. To this end, for any column j and row i such that $b_j \neq 1-b$, only one of the entries $A_{1-b}[i,j]$ or $B_{1-b}[i,j]$ is revealed (while the other entry is set to random, depending on the choice of c_i). This is because when $b_j = b$, the information regarding the $1-b$th matrices is available only from the rows being revealed. Next, note that $A_{1-b}[i,j], B_{1-b}[i,j]$ are individually distributed uniform at random, therefore $A_{1-b}[i,j] + B_{1-b}[i,j]$ is hidden. Now, since $b_j \neq 1-b$ for at least M values of j we conclude that at least M shares of x_{1-b} are hidden. Therefore, at most M shares can be recovered but M shares information theoretically hide x_{1-b}.

Therefore, we conclude that

Claim 5.3. *The following two distribution ensembles are computationally indistinguishable,*

$$\left\{\mathbf{View}_{\pi_{\mathrm{OT}},\mathcal{A}(z)}(n,(s_0,s_1),b)\right\}_{n\in\mathbb{N},s_0,s_1,b,z\in\{0,1\}^*} \overset{\mathrm{c}}{\approx} \left\{\mathbf{View}_{\mathcal{F}_{\mathrm{OT}},\mathcal{S}(z)}(n,(s_0,s_1),b)\right\}_{n\in\mathbb{N},s_0,s_1,b,z\in\{0,1\}^*}.$$

5.2 Obtaining Secure Two-Party Computation with $1/\mathrm{p}$-Security

Obtaining general secure two-party computation is carried out analogous to Protocol 3 by embedding the two-round protocol of [IKO$^+$11] within our second/third messages of our OT protocol. It follows just as before that we obtain a two-party computation protocol secure against malicious non-aborting adversaries.

Recall that, in our previous protocol, to achieve simulation when the receiver is corrupted, we consider a simulator that honestly generates the sender's messages with arbitrary inputs for the functionality being computed and then extracts the receiver's inputs to the OT by rewinding the WI-PoK. By relying on precisely the same strategy, we can obtain the receiver's inputs in this protocol and then complete the simulation by relying on the simulator for the malicious receiver in [IKO$^+$11] protocol.

To achieve simulation when the sender is corrupted, we combine the following two observations:

- First, using the approach from our previous protocol, it follows that whenever the simulator extracts the required trapdoor, it is possible to generate the OT part in the second message from the receiver in a way that it is identically distributed to the real receiver's message while at the same time extracting the sender's inputs to the OT. Furthermore, whenever the extraction of the sender's inputs is successful, we can rely on the simulation of the [IKO$^+$11] simulator in the $\mathcal{F}_{\mathrm{OT}}$-hybrid to complete the rest of the simulation.

- Second, we observe that, if the sender aborts before sending the third message, no extraction needs to be carried out since no inputs need to be feed to the $\mathcal{F}_{\mathrm{OT}}$-functionality.

We can now conclude that our simulation achieves $1/p$-security against malicious senders, by using the same two cases as we considered for the OT protocol based on the abort probability of the sender. More precisely,

Case: non-aborting probability of $\mathcal{A}$ is greater than $\frac{1}{pN}$. In this case, we know that except with probability $O(\frac{1}{p})$ the simulator extracts the required trapdoors and we achieve perfect simulation with probability at least $1 - O(\frac{1}{p})$.

Case: non-aborting probability of $\mathcal{A}$ is at most $\frac{1}{pN}$. If the non-aborting probability is smaller than $\frac{1}{pN}$ then the probability mass of aborting views is at least $1 - \frac{1}{pN} > 1 - \frac{1}{p}$ and since no extraction needs to be carried out we achieve $1/p$-indistinguishability.

Therefore, we have the following theorem:

Theorem 5.2. *Assuming the Decisional Diffie-Hellman problem is hard, there exists a four-round two-party secure computation protocol for any functionality that achieves security in the presence non-aborting adversary for one of the parties and $\frac{1}{p}$-security against the other party.*

6 On the Impossibility of Black-Box 3-Round 2PC with $1/\mathrm{p}$-Security

In this section, we show that constructing a 3-round secure two-party computation protocol (of single-output functionalities) with $1/p$-security against a corrupted receiver is impossible. More formally, we prove that

Theorem 6.1. *Unless $\mathsf{NP} \subseteq \mathsf{BPP}$, there exists no 3-round black-box construction of a secure two-party computation protocol with $1/p$-security against the receiver to realize arbitrary polynomial time computable functionalities.*

Proof: We rely on the following lemma, that essentially follows from the three-round lower bound for zero-knowledge (ZK) interactive proofs of Goldreich and Krawczyk [GK96].

Lemma 6.1. *Unless $\mathsf{NP} \subseteq \mathsf{BPP}$, there exists no black-box 3-round zero-knowledge interactive proofs for all of NP with $1/p$-security.*

Given the proof of Lemma 6.1, the theorem follows as a corollary. Consider an arbitrary NP-language L with witness relation R_L. Then, for any $x \in \{0,1\}^*$ consider the functionality $f_x : \{0,1\}^* \to \{0,1\}$ that on input w from P_1 outputs $R_L(x,w)$ to party P_2. In essence, a secure computation protocol for this functionality yields a zero-knowledge interactive proof. Moreover, it follows from the simulation-based definition of the $1/p$-security that if the original secure computation protocol is only $1/p$-secure, we get a zero-knowledge proof that is $1/p$-secure.

We now provide a brief overview of why Lemma 6.1 holds. We first recall the lower bound of Goldreich and Krawczyk. Suppose that there exists a 3-round ZK proof for an arbitrary NP language L. Consider a pseudo-random function family $F = \{f_n\}_{n\in\{0,1\}^*}$.[4] Then define a malicious verifier V_n^* that incorporates a function f_n from the PRF family F, and generates its second message of the ZK protocol by first generating randomness τ by applying f_n to the prover's first message and the running the honest verifier's code $\mathcal{V}$ with random tape set to τ. Consider the simulator $\mathcal{S}$ that simulates this family of malicious verifiers $\mathcal{V}_n^*$. The main idea here is that using the simulation $\mathcal{S}$ and $\mathcal{V}_n^*$ we can show that either $L \in \mathsf{BPP}$ or the interactive proof is not sound. On a high-level, from the pseudorandomness of the family F it follows that the real view generated by $\mathcal{V}_n^*$ is indistinguishable from the view that is generated by the real verifier $\mathcal{V}$. Hence, given input $x \in L$, $\mathcal{S}^{\mathcal{V}_n^*}$ produces a convincing view for the verifier with probability q negligibly close to 1. Moreover, on input $x \notin L$, $\mathcal{S}^{\mathcal{V}_n^*}$ either produces a convincing view or not. Concretely,

- If it does not produce a view with probability close to q for any $x \notin L$, then we can use $\mathcal{S}^{\mathcal{V}_n^*}$ as a BPP-decider for the language L by simply estimating the probability with which $\mathcal{S}^{\mathcal{V}_n^*}(x)$ outputs a convincing view.
- If it does produce a view with some probability close to q for some $x \notin L$, then we can construct a malicious prover $\mathcal{P}^*$ that convinces the honest verifier $\mathcal{V}$ of the statement x with non-negligible probability, which contradicts the soundness of the interactive proof. First, we observe that the view output by $\mathcal{S}^{\mathcal{V}_n^*}$ is indistinguishable from the output of $\mathcal{S}^{\widetilde{\mathcal{V}}}$ where $\widetilde{\mathcal{V}}$ uses a truly random function instead of a PRF function f_n to generate the randomness. Specifically, given input x, $\mathcal{P}^*$ internally

[4]For simplicity, we present the proof with PRF's. However, to get an unconditional result as stated in the lemma, we can rely on m-wise independent hash-function family where m is polynomially related to the expected running time of the simulator $\mathcal{S}$.

simulates $\mathcal{S}^{\widetilde{\mathcal{V}}}(x)$ by emulating the random function queries.[5] It then randomly chooses a session from the internal emulation and forwards the messages exchanged between $\mathcal{S}$ and $\widetilde{\mathcal{V}}$ to the external honest verifier. It follows that the view generated internally by $\mathcal{P}$ is identically distributed to the view generated by $\mathcal{S}^{\widetilde{\mathcal{V}}}(x)$. Furthermore, if the view output by $\mathcal{S}$ is the session forwarded externally to the honest verifier, then it implies that the external verifier essentially accepts.[6] Finally, suppose that the simulation runs in time T, then it follows that $\mathcal{P}^*$ guesses the correct session to forward outside with probability $\frac{q}{T}$. Therefore, it convinces the external verifier with probability close to $\frac{q}{T}$. Now, since T is some polynomial, it follows that $\mathcal{P}^*$ convinces $\mathcal{V}$ on an input $x \notin L$ with non-negligible probability and this violates soundness.

We now conclude the proof of the lemma by making the observation that even if the simulation was only $1/p$-indistinguishable, then $q = 1 - \frac{1}{p}$ and the success probability of $\mathcal{P}^*$ is still non-negligible. $\square$

References

[AL10] Yonatan Aumann and Yehuda Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. *J. Cryptology*, 23(2):281–343, 2010.

[Bea91] Donald Beaver. Foundations of secure interactive computing. In *CRYPTO*, pages 377–391, 1991.

[BOV07] Boaz Barak, Shien Jin Ong, and Salil P. Vadhan. Derandomization in cryptography. *SIAM J. Comput.*, 37(2):380–400, 2007.

[Can00] Ran Canetti. Security and composition of multi-party cryptographic protocols. *Journal of Cryptology*, 13:143–202, 2000.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO*, pages 174–187, 1994.

[CEvdG87] David Chaum, Jan-Hendrik Evertse, and Jeroen van de Graaf. An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In *EUROCRYPT*, pages 127–141, 1987.

[Cle86] Richard Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In *STOC*, pages 364–369, 1986.

[DFH12] Ivan Damgård, Sebastian Faust, and Carmit Hazay. Secure two-party computation with low communication. In *TCC*, pages 54–74, 2012.

[DN07] Cynthia Dwork and Moni Naor. Zaps and their applications. *SIAM J. Comput.*, 36(6):1513–1543, 2007.

[Fis01] Marc Fischlin. Trapdoor commitment schemes and their applications. *Ph.D. Thesis*, 2001.

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *STOC*, pages 416–426, 1990.

[5] Namely, on any input query to the random function, $\mathcal{P}^*$ checks if the query has already been asked and produces the same answer in this case. Otherwise, it samples and feeds a uniform output and records the query/answer pair.

[6] This is not entirely accurate as $\mathcal{P}^*$ does not know the actual randomness used by the external verifier since this is a private-coin protocol. Nevertheless, it is possible to formally prove that conditioned on $\mathcal{P}^*$ guessing correctly, $\mathcal{P}^*$ convinces the external verifier with probability equal to the probability $\mathcal{S}$ outputs a convincing view in the internal emulation, i.e. close to q.

[Gam85] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[GK96] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM J. Comput.*, 25(1):169–192, 1996.

[GK10] S. Dov Gordon and Jonathan Katz. Partial fairness in secure two-party computation. In *EUROCRYPT*, pages 157–176, 2010.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *J. Cryptology*, 7(1):1–32, 1994.

[Gol04] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, New York, NY, USA, 2004.

[Hai08] Iftach Haitner. Semi-honest to malicious oblivious transfer - the black-box way. In *TCC*, pages 412–426, 2008.

[HIK$^+$11] Iftach Haitner, Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. Black-box constructions of protocols for secure computation. *SIAM J. Comput.*, 40(2):225–266, 2011.

[HK07] Omer Horvitz and Jonathan Katz. Universally-composable two-party computation in two rounds. In *CRYPTO*, pages 111–129, 2007.

[HK12] Shai Halevi and Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. *J. Cryptology*, 25(1):158–193, 2012.

[IKKPC15] Yuval Ishai, Ranjit Kumaresan, Eyal Kushilevitz, and Anat Paskin-Cherniavsky. Secure computation with minimal interaction, revisited. In *CRYPTO*, page To appear, 2015.

[IKO$^+$11] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, and Amit Sahai. Efficient non-interactive secure computation. In *EUROCRYPT*, pages 406–425, 2011.

[Kat12] Jonathan Katz. Which languages have 4-round zero-knowledge proofs? *J. Cryptology*, 25(1):41–56, 2012.

[KO04] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In *CRYPTO*, pages 335–354, 2004.

[Lin01] Yehuda Lindell. Parallel coin-tossing and constant-round secure two-party computation. In *CRYPTO*, pages 171–189, 2001.

[MNS09] Tal Moran, Moni Naor, and Gil Segev. An optimally fair coin toss. In *TCC*, pages 1–18, 2009.

[MPR06] Silvio Micali, Rafael Pass, and Alon Rosen. Input-indistinguishable computation. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 367–378, 2006.

[MR91] Silvio Micali and Phillip Rogaway. Secure computation (abstract). In *CRYPTO*, pages 392–404, 1991.

[NP01] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *SODA*, pages 448–457, 2001.

[ORS15] Rafail Ostrovsky, Silas Richelson, and Alessandra Scafuro. Round-optimal black-box two-party computation. *IACR Cryptology ePrint Archive*, 2015:553, 2015.

[Ped91] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO*, pages 129–140, 1991.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[Yao82] Andrew C. Yao. Theory and application of trapdoor functions. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, SFCS '82, pages 80–91, Washington, DC, USA, 1982. IEEE Computer Society.

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

A Preliminaries – Appendix

A.1 Public Key Encryption Schemes (PKE)

We specify the definitions of public key encryption and IND-CPA.

Definition A.1 (PKE). *We say that* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is a* public key encryption scheme *if* $\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$ *are polynomial-time algorithms specified as follows:*

- Gen*, given a security parameter n (in unary), outputs keys* $(\mathrm{PK}, \mathrm{SK})$*, where* PK *is a public key and* SK *is a secret key. We denote this by* $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Gen}(1^n)$.
- Enc*, given the public key* PK *and a plaintext message m, outputs a ciphertext c encrypting m. We denote this by $c \leftarrow \mathsf{Enc}_{\mathrm{PK}}(m)$; and when emphasizing the randomness r used for encryption, we denote this by $c \leftarrow \mathsf{Enc}_{\mathrm{PK}}(m; r)$.*
- Dec*, given the public key* PK*, secret key* SK *and a ciphertext c, outputs a plaintext message m s.t. there exists randomness r for which $c = \mathsf{Enc}_{\mathrm{PK}}(m; r)$ (or $\perp$ if no such message exists). We denote this by $m \leftarrow \mathsf{Dec}_{\mathrm{PK},\mathrm{SK}}(c)$.*

For a public key encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and a non-uniform adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we consider the following *IND-CPA game* denoted by $\textsc{Adv}_{\Pi,\mathcal{A}}(n)$:

$$
\begin{aligned}
&(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Gen}(1^n).\\
&(m_0, m_1, \mathit{history}) \leftarrow \mathcal{A}_1(\mathrm{PK}), \text{ s.t. } |m_0| = |m_1|.\\
&c \leftarrow \mathsf{Enc}_{\mathrm{PK}}(m_b), \text{ where } b \leftarrow_R \{0,1\}.\\
&b' \leftarrow \mathcal{A}_2(c, \mathit{history}).\\
&\text{Return } 1 \text{ if } b' = b, \text{ and } 0 \text{ otherwise.}
\end{aligned}
$$

Definition A.2 (IND-CPA)**.** *A public key encryption scheme* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *has* indistinguishable encryptions under chosen plaintext attacks *(IND-CPA), if for every non-uniform adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *there exists a negligible function* negl *such that*

$$\Pr[\textsc{Adv}_{\Pi,\mathcal{A}}(n) = 1] \leq \frac{1}{2} + \mathsf{negl}(n)$$

where the probability is taken over the random coins used by $\mathcal{A}$*, as well as the random coins used in the experiment.*

A.1.1 The El Gamal PKE

A useful implementation of homomorphic PKE is the El Gamal [Gam85] scheme that is multiplicatively homomorphic. In this paper we exploit the additive variation. Let $\mathbb{G}$ be a group of prime order p in which DDH is hard. Then the public key is a tuple PK $= \langle \mathbb{G}, p, g, h \rangle$ and the corresponding secret key is SK $= s$, s.t. $g^s = h$. Encryption is performed by choosing $r \leftarrow \mathbb{Z}_p$ and computing $\mathsf{Enc}_{\mathrm{PK}}(m; r) = \langle g^r, h^r \cdot m \rangle$. Decryption of a ciphertext $C = \langle \alpha, \beta \rangle$ is performed by computing $m = \beta \cdot \alpha^{-s}$ and then finding m by running an exhaustive search.

A.2 Knowledge Extraction

In this paper we are interested in witness indistinguishable and zero-knowledge proofs that are proofs of knowledge (PoK) which imply the existence of a knowledge extractor that extracts the witness w used by the prover.

Definition A.3. *Let* R *be a binary relation and* $\kappa \to [0, 1]$. *We say that an interactive function* $\mathcal{V}$ *is* a knowledge verifier for the language L with knowledge error κ *if the following two conditions hold:*

Non-triviality: *There exists an interactive machine* $\mathcal{P}$ *such that for every* (x, w) *such that* w *is a witness for* $x \in L$*, all possible interactions of* $\mathcal{V}$ *with* $\mathcal{P}$ *on common input* x *and auxiliary input* w *are accepting.*

Validity (with error κ): *There exists a polynomial* $q(\cdot)$ *and a probabilistic oracle machine* K *such that for every interactive function* $\mathcal{P}$*, every* $x \in L$*, and every machine* K *satisfies the following condition:*

> *Denote by* $p(x, y, r)$ *the probability that the interactive machine* $\mathcal{V}$ *accepts, on input* x*, when interacting with the prover specified by* $\mathcal{P}_{x,y,r}$ *that uses randomness* r *(where the probability is taken over the coins of* $\mathcal{V}$*). If* $p(x, y, r) > \kappa(|x|)$*, then, on input* x *and with access to oracle* $\mathcal{P}_{x,y,r}$*, machine* K *outputs a witness* s *for* $x \in L$ *within an expected number of steps bounded by*
>
> $$\frac{q(|x|)}{p(x, y, r) - \kappa(|x|)}$$
>
> *The oracle machine* K *is called a universal knowledge extractor.*

It is known that any Σ-protocol is a WI-PoK. One such example is the protocol for proving the knowledge of Hamiltonian cycle in a graph, which is an NP-complete problem.

A.3 Secure Two-Party Computation

We briefly present the standard definition for secure multiparty computation and refer to [Gol04, Chapter 7] for more details and motivating discussions. A two-party protocol problem is cast by specifying a random process that maps pairs of inputs to pairs of outputs (one for each party). We refer to such a process as a *functionality* and denote it $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$, where $f = (f_1, f_2)$. That is, for every pair of inputs (x, y), the output-vector is a random variable $(f_1(x, y), f_2(x, y))$ ranging over pairs of strings where P_1 receives $f_1(x, y)$ and P_2 receives $f_2(x, y)$. We use the notation $(x, y) \mapsto (f_1(x, y), f_2(x, y))$ to describe a functionality. We prove the security of our protocols in the settings of *malicious* computationally bounded adversaries. Security is analyzed by comparing what an adversary can do in a *real* protocol execution to what it can do in an *ideal* scenario. In the ideal scenario, the computation involves an incorruptible *trusted third party* to whom the parties send their inputs. The trusted party computes the functionality on the inputs and returns to each party its respective output. Informally, the protocol is secure if any adversary interacting in the real protocol (i.e., where no trusted third party exists) can do no more harm than what it could do in the ideal scenario. In this paper we follow the $\frac{1}{p}$-secure computation definition from [GK10] which presented a simulation based definition for which the difference between the real and the simulated distributions differ within $\frac{1}{p}$.

Execution in the ideal model. In an ideal execution, the parties submit inputs to a trusted party, that computes the output. An honest party receives its input for the computation and just directs it to the trusted party, whereas a corrupted party can replace its input with any other value of the same length. Since we do not consider fairness, the trusted party first sends the outputs of the corrupted parties to the adversary, and the adversary then decides whether the honest parties would receive their outputs from the trusted party or an *abort* symbol $\perp$. Let f be a two-party functionality where $f = (f_1, f_2)$, let $\mathcal{A}$ be a non-uniform probabilistic polynomial-time algorithm, and let $I \subset [2]$ be the set of corrupted parties (either P_1 is corrupted or P_2 is corrupted or neither). Then, the *ideal execution of* f on inputs (x, y), auxiliary input z to $\mathcal{A}$ and security parameter n, denoted $\mathbf{IDEAL}_{f,\mathcal{A}(z),I}(n, x, y)$, is defined as the output pair of the honest party and the adversary $\mathcal{A}$ from the above ideal execution.

Execution in the real model. In the real model there is no trusted third party and the parties interact directly. The adversary $\mathcal{A}$ sends all messages in place of the corrupted party, and may follow an arbitrary polynomial-time strategy. The honest parties follow the instructions of the specified protocol π.

Let f be as above and let π be a two-party protocol for computing f. Furthermore, let $\mathcal{A}$ be a non-uniform probabilistic polynomial-time algorithm and let I be the set of corrupted parties. Then, the *real execution of* π on inputs (x, y), auxiliary input z to $\mathcal{A}$ and security parameter n, denoted $\mathbf{REAL}_{\pi,\mathcal{A}(z),I}(n, x, y)$, is defined as the output vector of the honest parties and the adversary $\mathcal{A}$ from the real execution of π.

Security as emulation of a real execution in the ideal model. Having defined the ideal and real models, we can now define security of protocols. Loosely speaking, the definition asserts that a secure party protocol (in the real model) emulates the ideal model (in which a trusted party exists). This is formulated by saying that adversaries in the ideal model are able to simulate executions of the real-model protocol.

Definition A.4. *Let f and π be as above. Protocol π is said to* securely compute f with abort in the presence of malicious adversaries *if for every non-uniform probabilistic polynomial-time adversary $\mathcal{A}$ for the real model, there exists a non-uniform probabilistic polynomial-time adversary $\mathcal{S}$ for the ideal model, such that for every $I \subset [2]$,*

$$\left\{\mathbf{IDEAL}_{f,\mathcal{S}(z),I}(n, x, y)\right\}_{n\in\mathbb{N},x,y,z\in\{0,1\}^*} \overset{1/\text{p}}{\approx} \left\{\mathbf{REAL}_{\pi,\mathcal{A}(z),I}(n, x, y)\right\}_{n\in\mathbb{N},x,y,z\in\{0,1\}^*}$$

where n is the security parameter.

The $\mathcal{F}$-hybrid model. In order to construct some of our protocols, we will use secure two-party protocols as subprotocols. The standard way of doing this is to work in a "*hybrid model*" where parties both interact with each other (as in the real model) and use trusted help (as in the ideal model). Specifically, when constructing a protocol π that uses a subprotocol for securely computing some functionality $\mathcal{F}$, we consider the case that the parties run π and use "ideal calls" to a trusted party for computing $\mathcal{F}$. Upon receiving the inputs from the parties, the trusted party computes $\mathcal{F}$ and sends all parties their output. Then, after receiving these outputs back from the trusted party the protocol π continues. Let $\mathcal{F}$ be a functionality and let π be a two-party protocol that uses ideal calls to a trusted party computing $\mathcal{F}$. Furthermore, let $\mathcal{A}$ be a non-uniform probabilistic polynomial-time algorithm. Then, the *$\mathcal{F}$-hybrid execution of* π on inputs (x, y), auxiliary input z to $\mathcal{A}$ and security parameter n, denoted $\mathsf{hyb}_{\pi^{\mathcal{F}},\mathcal{A}(z)}(n, x, y)$, is defined as the output vector of the honest parties and the adversary $\mathcal{A}$ from the hybrid execution of π with a trusted party computing $\mathcal{F}$. By the composition theorem of [Can00] any protocol that securely implements $\mathcal{F}$ can replace the ideal calls to $\mathcal{F}$.