Proceedings Paper:
Abdolmaleki, B., Badrinarayanan, S., Fernando, R. et al. (3 more authors) (2023) Two-round concurrent 2PC from sub-exponential LWE. In: Advances in Cryptology – ASIACRYPT 2023: 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4–8, 2023, Proceedings, Part I. 29th International Conference on the Theory and Application of Cryptology and Information Security, 04-08 Dec 2023, Guangzhou, China. Lecture Notes in Computer Science, LNCS 14438 . Springer Nature Singapore , pp. 71-105. ISBN 9789819987207

https://doi.org/10.1007/978-981-99-8721-4_3

Two-Round Concurrent 2PC from Sub-Exponential LWE

Behzad Abdolmaleki[1], Saikrishna Badrinarayanan[2⋆], Rex Fernando[3⋆], Giulio Malavolta[4,5], Ahmadreza Rahimi[5], and Amit Sahai[6]

[1] University of Sheffield, UK
behzad.abdolmaleki@sheffield.ac.uk
[2] LinkedIn, USA
bsaikrishna7393@gmail.com
[3] Carnegie Mellon University, USA
rex1fernando@gmail.com
[4] Bocconi University, Italy
[5] Max Planck Institute for Security and Privacy, Germany
{giulio.malavolta, ahmadreza.rahimi}@mpi-sp.org
[6] UCLA, USA
sahai@cs.ucla.edu

Abstract. Secure computation is a cornerstone of modern cryptography and a rich body of research is devoted to understanding its round complexity. In this work, we consider two-party computation (2PC) protocols (where both parties receive output) that remain secure in the realistic setting where many instances of the protocol are executed in parallel (concurrent security). We obtain a two-round *concurrent-secure* 2PC protocol *based on a single, standard, post-quantum* assumption: The subexponential hardness of the learning-with-errors (LWE) problem. Our protocol is in the plain model, i.e., it has no trusted setup, and it is secure in the super-polynomial simulation framework of Pass (EUROCRYPT 2003). Since two rounds are minimal for (concurrent) 2PC, this work resolves the round complexity of concurrent 2PC from standard assumptions.

As immediate applications, our work establishes feasibility results for interesting cryptographic primitives, such as the first two-round password authentication key exchange (PAKE) protocol in the plain model and the first two-round concurrent secure computation protocol for quantum circuits (2PQC).

1 Introduction

Secure computation is a fundamental primitive in cryptography which allows two or more parties, all of whom have private inputs, to collectively compute some function over their inputs securely without revealing the inputs themselves. In recent years, significant attention has been devoted to the round-complexity of

⋆ Part of the work was done while the author was affiliated with UCLA.

secure computation in the setting of two parties, as well as in the multi-party setting (MPC). This has culminated in recent work of [37, 4, 22, 10, 45, 29], which give protocols that run in four rounds, known to be the least amount of rounds possible for full security in the plain model[7].

The results above achieve security in the *standalone* setting, where all parties are assumed to participate in only one instance of the protocol.

The concurrent setting. A more realistic setting allows parties to participate *concurrently* in arbitrarily many instances. Unfortunately, Barak, Prabhakaran and Sahai [13] show that achieving the standard definition of concurrent security is impossible *in any rounds* in the plain model, without a trusted setup. In an effort to overcome the above mentioned impossibility results, many recent works have focused on proving concurrent security for two-party computation (2PC) in alternative models, e.g., in the bounded concurrent model [64], in the multiple ideal-query model [42], and for input-indistinguishable computation [60].

One standard relaxation of simulation security, which is widely used to circumvent many lower-bound results, is the notion of *super-polynomial simulation*, or SPS [63]. With this notion, for any real-world adversary, we require an ideal-world simulator that runs in super-polynomial time. More precisely, in this scenario, the simulator in the ideal world is allowed to run in (fixed) super-polynomial time. Informally, the SPS security guarantees that any polynomial-time attack in the real execution can also be mounted in the ideal world execution, albeit in super-polynomial time. This is directly applicable in settings where ideal world security is guaranteed statistically or information-theoretically and it is known to imply input-indistinguishable computation [60]. There has been a fruitful line of research devoted to understanding the power of SPS security for secure computations in the concurrent setting [59, 27, 35, 54, 65, 53, 52] [44, 36, 11].

The round complexity. In the concurrent setting, a series of works [35, 53] constructed constant-round protocols (approximately 20 rounds) in the simultaneous message exchange model. Later, Garg *et al.* [36] decreased the round complexity to 5 rounds with SPS security from standard sub-exponential assumptions. In 2017, the work of Badrinarayanan et al. [11] used this notion to circumvent both the impossibility of concurrent MPC and the four-round lower-bound, giving a protocol that works in three rounds and satisfies concurrent security. For several years, this result was the best-known result regarding the round complexity of MPC in the plain model. Until very recently, no two-round protocols were known. This was the case even in the restricted setting of two-party computation (where both parties receive output).

Recently two new works improved the state of the art in this area:

[7] It is also known how to achieve two-round MPC that satisfies a much weaker notion of *semi-malicious* security, where the adversary is assumed to follow the honest protocol specification. [38] Alternately, achieving full security in two rounds is possible if we allow for a trusted setup. In this paper, we focus on achieving full malicious security in the plain model, without setup.

– The work of [1] gave a two-round MPC protocol for general functionalities which achieves standalone security in the plain model without setup and with a super-polynomial simulator, assuming subexponential non-interactive witness-indistinguishable arguments, the subexponential SXDH assumption, and the existence of a special type of non-interactive non-malleable commitment.[8]
– The work of [33] gave a concurrent, highly-reusable[9] two-round MPC protocol for general functionalities, assuming subexponential quantum hardness of the learning-with-errors (LWE) problem, subexponential classical hardness of SXDH, the existence of a subexponentially-secure (classically-hard) indistinguishability obfuscation (iO) scheme, and time-lock puzzles.

Assumptions for two-party secure computation. The goal of our work is to focus on secure computation in the two-party setting and to explore the assumptions under which two-round secure protocols are possible. Even in this more specific setting, the two above results are the only known protocols that achieve two-round protocols for general two-party functionalities.[10] Both of the previously mentioned works on two-round protocols use powerful primitives which are only known from strong assumptions. More specifically, the work of [1] requires a strong version of non-interactive non-malleable commitments, which are only known from strong, non-standard assumptions, such as adaptive one-way functions [62], or keyless hash functions along with a subexponential variant of the "hardness amplifiability" assumption of [18]. The work of [33] is able to avoid using these strong commitments, instead using (a modified version of) the one-round NMC of [50], which relies on the existence of sub-exponential indistinguishability obfuscation (iO).

We briefly discuss the assumptions under which iO exists. Our understanding of these assumptions has vastly improved in recent years, culminating in the work of [49, 48], which showed that iO can be built on well-founded assumptions, namely hardness of LPN over $\mathbb{F}_p$, hardness of DLIN, and the existence of PRGs in NC^0. However, our understanding of the assumptions necessary for *quantum-secure* iO is much less stable. We note that besides the above-mentioned work, all other constructions of iO rely on ad-hoc hardness assumptions which were specifically invented for the purpose of proving the security of iO [34, 23, 67, 7, 12, 28, 30, 61, 57, 6, 56, 5, 2, 47, 15, 3, 20, 39, 21, 70]. Although some of the most recent of these constructions rely on lattice-based assumptions which ostensibly could be quantum-secure [39, 21, 70], there are already preliminary attacks on some versions of these new assumptions [46]. Thus, in this setting, our understanding is much more limited than in the classical case.

[8] The protocol of [1] is given in the form of a compiler that transforms a two-round semi-malicious-secure MPC protocol into a malicious-secure one.

[9] See [33] for the exact definition of reusability obtained.

[10] If we restrict ourselves to functionalities where only one party receives output, then it is known how to achieve two-round secure computation from much simpler assumptions [9], in the setting of standalone security.

In addition to iO, both of the constructions of the two-round MPC above use other assumptions (i.e, SXDH) which, while standard, are quantum-broken. With all of this in mind, it is interesting to ask the following question:

Can we achieve two-round concurrently secure two-party computation under simple, post-quantum assumptions, in the plain model?

As mentioned above, this question is interesting even if we restrict ourselves to the case of two parties, since up to this point the only known results even in this subcase are the two discussed above, which both require strong, potentially quantum-unsafe assumptions.

1.1 Our Contributions

In this work, we make a significant process in answering the above question. In this particular case, we show how to build a two-round, *concurrent-secure*, two-party secure computation protocol *based on a single, standard, post-quantum* assumption, namely sub-exponential the hardness of the learning-with-errors (LWE) problem. We state our main theorem now.

Theorem 1. *Assuming the sub-exponential hardness of the learning-with-errors (LWE) problem, there exists a two-round two-party computation protocol for any polynomial-time functionality f where both parties receive outputs, in the plain model with super-polynomial simulation.*

We note that our protocol is the first two-round concurrent-secure 2PC the protocol that does not require the existence of a one-round non-malleable commitment (NMC). Instead, we are able to use the two-round NMCs of [51], which is instantiable from sub-exponential LWE. Our protocol is also the first such protocol that does not require the existence of non-interactive witness indistinguishable arguments or time-lock puzzles. Here, we briefly mention two of the applications of our protocol.

Application: Round-optimal PAKE. In a password-authenticated key exchange (PAKE) [17] protocol, two users hold passwords $(\mathbf{x}_1, \mathbf{x}_2)$ and want to exchange a high-entropy secret if $\mathbf{x}_1 = \mathbf{x}_2$, otherwise, they learn nothing about the other user's inputs. Our concurrent 2PC protocol directly yields the first two-round PAKE scheme in the plain model, resolving a long-standing open problem in the area. Our protocol achieves the standard game-based security notion, as defined by Bellare *et al.* [16].

Application: Concurrent quantum computation. As another application of our protocol, we show how it immediately yields the first concurrent 2PC for quantum functionalities (in the plain model) with classical inputs and outputs. In fact, we show that our classical 2PC provides us with the necessary building block to instantiate the recent compiler of Bartusek *et al.* [14], which we then show how to lift to the concurrent settings.

1.2 Technical Overview

To introduce the techniques used in our concurrent 2PC construction, we start by summarizing a discussion in the work of [11], which gives an intuition for why two-round secure computation protocols seem difficult to achieve. In particular, they argue that such a protocol seems to necessarily imply non-interactive non-malleable commitments (NMCs).

Difficulties in constructing concurrent 2PC. We focus our summary on the case of two parties since our paper addresses this case. The authors of [11] note that any such two-round 2PC protocol should have some sort of input commitment in the first round, and then the second round should be used to compute the output. They then make the following important observation: Since we are working in the SPS model without setup, zero knowledge requires at least two rounds. This means that an honest party must send its second message *without knowing if the adversary's first-round message is honest.* The example given in [11] to illustrate this is as follows. Consider a case where the honest party's input is x, and where there is a "rushing" adversary which waits to send its first message until after seeing the honest party's first message. If this adversary "mauls" the other party's message and sends a first-round message which also encodes x, then the honest party cannot detect this before sending out its second message. Ostensibly, this would cause both the honest party and the adversary to learn $f(x, x)$, thus breaking SPS security of the protocol. Because of this, it seems at first glance that non-interactive NMCs are necessary to prevent such "mauling" attacks in two-round protocols.

Avoiding non-interactive NMCs. As discussed earlier, we want to avoid non-interactive NMCs, since contrary to two-round NMCs all non-interactive constructions require strong assumptions. To do so, we must understand why the intuition above is incorrect. One implicit assumption we have made in this argument is that the adversary always learns the same output that the honest party learns after the second round. This is indeed the case in *public-output* protocols, where anyone can compute the output given just the transcript of the protocol and no other information. However, what about the case of *private-output* protocols, where each party must use private information in order to reconstruct the output? In such protocols, when proving security, it is easier to separate the adversary's output (or more generally, its view) from the output of the honest party. Notice that in the example above, the adversary sends its first message without even knowing the input x which it is encoding. (In order to prove security, at the very least, we must assume the honest party's first-round message does not leak its input.) If we could somehow guarantee that the adversary can only unlock the protocol output if it knows its own input, this would prevent the adversary from learning $f(x, x)$.

Our approach. We use all these observations in order to obtain our construction. We will use four main tools in our construction: (1) a two-round non-malleable commitment, (2) a two-round statistically-sender-private oblivious transfer protocol (SSP OT), (3) a two-round strong SPS zero-knowledge

protocol, and (4) garbled circuits. We will require each party P_i to publish two different types of commitments to its input, one using the NMC and the other using an OT_1 message. Roughly, the OT_1 message will be used by party P_i in reconstructing its own output and the NMC will be used to help P_{1-i} to reconstruct its output. Crucially, we will show that the non-malleability of the NMC *is not needed for the privacy of the protocol,* i.e., it is not needed to prevent the adversary from learning $f(x, x)$. Rather, it is only needed *in order to prevent the honest party from learning "mauled" outputs such as* $f(x, x)$. Note that although the (rushing) adversary's output must be decided before receiving the adversary's second-round NMC message, the honest party's output can be decided after seeing the entire transcript of the protocol. Thus two-round non-malleability is easily sufficient to prevent mauling in terms of the honest party's output. How do we prevent the adversary from learning $f(x, x)$, then? At a high level, we rely on the SSP oblivious transfer, which satisfies exactly the property we hinted at above: an adversary can only unlock the protocol output if it knows the input of its OT_1 message.

Putting these ideas into practice involves several technical issues. We discuss a few here. One obvious issue is that we must somehow connect the NMC with the OT_1. Otherwise, it would be possible for the adversary to learn $f(x_1, y)$ whereas the honest party learns $f(x_2, y)$. To do this, we observe that it is possible to construct a *simultaneous-message* two-round NMC scheme, where both the committer and receiver send a message in the first round, and where the first round is *binding*. That is, the first round defines a unique x such that after the second round, either the transcript commits non-malleably to x, or the transcript is invalid and cannot be opened. With that in mind, we require that in addition to committing to its input in its OT_1 message, P_i must also commit to the randomness used for its NMC1 message. P_{1-i} then can construct its garbled circuit to only reveal the output if this randomness is correct.

Security analysis. Although this solution to the problem seems simple, it introduces some subtleties to the proof of security. One such subtlety is in the hybrid order when moving from the real world to the ideal world. Namely, we must switch the OT_1 of the honest party to be an OT_1 of 0 before we switch the honest party's NMC to commit to 0. Since in the real world, the honest party gets its output by opening the adversary's OT_2 message, and in the ideal world it gets its output by extracting the adversary's NMC, this causes there to be several intermediate hybrids where there is no way for the honest party to compute its output. We must carefully prove that during these hybrids, although the honest party cannot compute its output, the output is well-defined and does not change in any computationally distinguishable way across these hybrids. We refer to Section 3.3 for details.

One other issue arises in the use of zero-knowledge to prove the honest generation of the garbled circuits with respect to the inputs committed to in the NMCs. That is, at some point during the hybrids, we must switch the honest party from using real proofs to using the SPS zero-knowledge simulator. Once we do this, we must somehow guarantee that the adversary (who now is re-

ceiving simulated proofs) cannot somehow use them to behave dishonestly. The work of [11] offers techniques to solve this issue, which is highly related to the notion of *simulation-soundness* [68]. Their main idea involves using *strong* SPS-zero-knowledge arguments in conjunction with non-malleable commitments. In strong SPS-ZK arguments, the zero-knowledge the property holds even against adversaries who are powerful enough to run the simulator. We are able to use the same techniques, although they require careful work to adapt to our setting and security proof. We again refer to Section 3.3 for more details on this and other technical issues.

Applications. We highlight two applications of our newly developed concurrent 2PC protocol.

(i) Round-optimal PAKE (Appendix B): In Password-Authenticated Key-Exchange (PAKE) two parties want to exchange a session key if their (low-entropy) passwords match. This functionality is a special case of general 2PC, so it is clear that any 2PC protocol immediately yields a PAKE scheme. However, the de-facto security notion for PAKE [16] models security in the presence of concurrent sessions. Thus, only concurrently secure 2PC properly generalizes PAKE to all functionalities. As a corollary of our main theorem, we obtain the first round-optimal PAKE without a trusted setup. This settles a long-standing question in the area.

(ii) Quantum computation (Section 5): Observe that we can lift our result to the quantum setting by plugging in our concurrent 2PC protocol in the construction of [14]. At the high level, the protocol of [14] converts any quantum-secure 2PC to a quantum 2PC, where parties wish to securely compute a quantum circuit on their input. In each round, parties compute the encoding of their quantum inputs, and in parallel, they run a classical concurrent secure 2PC to compute the classical description of the quantum garbled circuit. In the end of the second round, the sender can evaluate the circuit and get the output. [14] requires a 2PC with a straight-line simulator (which we can instantiate with our protocol) and a quantum garbling scheme [24]. One subtlety in the proof is that we need to adjust the security parameter for the quantum garbled circuits, as our simulator has sub-exponential run-time (i.e., we use complexity leveraging).

2 Preliminaries

In the following, we write $T_1 \ll T_2$ for functions T_1 and T_2 if for all polynomials p, $p(T_1(\lambda))$ is asymptotically smaller than $T_2(\lambda)$. We denote with $\mathcal{G}(x;r)$ the execution of a probabilistic algorithm $\mathcal{G}$, where x is the input to the algorithm and r is the string of random coins. When we do not need to explicitly deal with the random coins of $\mathcal{G}$, we write $\mathcal{G}(x)$ and assume that the coins r are chosen uniformly at random. Additional definitions are given in Appendix A.

2.1 Two-Round SPS Strong Zero Knowledge

We define the notion of two-round strong zero knowledge with super-polynomial simulation first given in [51]. Here *strong* means that the zero-knowledge prop-

erty holds even against adversaries which themselves are strong enough to run the simulator.

We consider zero-knowledge protocols with the following syntax. All algorithms below are polynomial-time.

- $\mathsf{ZK}_1(1^\lambda; r) \to \mathsf{zk}_1$ takes as input the security parameter 1^λ along with randomness r and produces the verifier's message.
- $\mathsf{ZK}_2(1^\lambda, x, w, \mathsf{zk}_1; r') \to \mathsf{zk}_2$ takes as input security parameter, the statement x and the witness w along with the verifier's message and randomness r' and produces the prover's message.
- $\mathsf{ZK}_{\mathsf{verify}}(x, \mathsf{zk}_2, r) \to 0/1$ is a deterministic algorithm which takes the statement x along with the prover's message and the randomness used to generate the verifier's message and accepts or rejects.

Definition 1 ($(T_{\mathsf{sound}}, T_{\mathsf{Sim}}, T_{\mathsf{zk}}, T_{\mathcal{L}}, \epsilon_{\mathsf{sound}}, \epsilon_{\mathsf{zk}})$-SPSS Zero-Knowledge Arguments).

Let $\mathcal{L}$ be a language in NP which is decidable in time $T_{\mathcal{L}}$, with a polynomial-time computable relation $\mathcal{R}_{\mathcal{L}}$. Let T_{sound}, T_{Sim}, T_{zk} be superpolynomial functions where $T_{\mathsf{sound}} \ll T_{\mathsf{Sim}} \ll T_{\mathsf{zk}} \ll T_{\mathcal{L}}$, and ϵ_{zk}, $\epsilon_{\mathsf{sound}}$ negligible functions. A protocol between a prover P and a verifier V is a $(T_{\mathsf{sound}}, T_{\mathsf{Sim}}, T_{\mathsf{zk}}, T_L, \epsilon_{\mathsf{sound}}, \epsilon_{\mathsf{zk}})$-strong zero-knowledge argument for $\mathcal{L}$ if it satisfies the following properties:

- ***Perfect Completeness.*** *For every security parameter 1^λ and NP statement x and witness w where $(x, w) \in R_{\mathcal{L}}$, it holds that*

$$\Pr\left[\mathsf{ZK}_{\mathsf{verify}}(x, \mathsf{zk}_2, r)\right] = 1,$$

  where $\mathsf{zk}_1 \leftarrow \mathsf{ZK}_1(1^\lambda; r)$ and $\mathsf{zk}_2 \leftarrow \mathsf{ZK}_2(1^\lambda, x, w, \mathsf{zk}_1; r')$ and the probability is taken over the randomness of r and r'.
- *$(T_{\mathsf{sound}}, \epsilon_{\mathsf{sound}})$-**Adaptive Soundness.** For every polynomial $p(\lambda)$ and every prover P^* that works in time T_{sound} and is given 1^λ and an honest verifier message zk_1; If P^* chooses an input length 1^p for some polynomial $p \in \mathsf{poly}(\lambda)$, and then chooses $x \in \{0,1\}^p \setminus \mathcal{L}$ and outputs (x, zk_2), it holds that*

$$\Pr\left[\mathsf{ZK}_{\mathsf{verify}}(x, \mathsf{zk}_2, r) = 1\right] \leq \epsilon_{\mathsf{sound}}(\lambda),$$

  where r is the randomness used to generate zk_1 and the probability is over the random coins of V.
- *$(T_{\mathsf{Sim}}, T_{\mathsf{zk}}, \epsilon_{\mathsf{zk}})$-**Strong Zero-Knowledge.** There exists a (uniform) simulator Sim which runs in time T_{Sim} which takes as input the round-one transcript zk_1 and a statement x such that the following holds. Consider an adversary V^* which runs in time T_{zk} that takes as input 1^λ and advice z and outputs a verifier's first round message zk_1^*. Then, for all $(x, w) \in R_{\mathcal{L}}$, distinguishers $\mathcal{D}$ which run in time T_{zk}, and advice z,*

$$\left|\Pr\left[\mathcal{D}(x, z, r, \mathsf{ZK}_2(1^\lambda, x, w, \mathsf{zk}_1^*)) = 1\right] - \Pr\left[\mathcal{D}(x, z, r, \mathsf{Sim}(1^\lambda, x, \mathsf{zk}_1^*)) = 1\right]\right| < \epsilon_{\mathsf{zk}}(\lambda),$$

  where r is the private randomness of V^*.

2.2 Two-Round Statistically-Sender-Private Oblivious Transfer

We give the formal definition of two-round oblivious transfer, where the receiver's security is computational, and there exists a (possibly computationally unbounded) extractor for the receiver's first-round message such that statistical security holds for the sender. The Oblivious Transfer scheme consists of the following polynomial-time algorithms:

- $\mathsf{OT}_1(1^\lambda, b; r) \to \mathbf{ot}_1$: The receiver's OT_1 algorithm takes a choice bit b and produces the receiver's OT message.
- $\mathsf{OT}_2(1^\lambda, \ell_0, \ell_1, \mathbf{ot}_1; r') \to \mathbf{ot}_2$: The sender's OT_2 algorithm takes a pair of strings to choose from along with the receiver's OT message and produces the sender's OT message.
- $\mathsf{OT}_3(\mathbf{ot}_2; r) \to \ell_b$: The receiver's OT_3 takes the sender's OT message and outputs ℓ_b.

Definition 2. *A tuple $(\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$ is a $(T_R, \epsilon_R, \epsilon_S)$-statistically-sender-private oblivious transfer algorithm if the following properties hold:*

- ***Correctness.*** *For all λ, b, ℓ_0, ℓ_1,*

$$\Pr\left[\mathsf{OT}_3(\mathbf{ot}_2; r) = \ell_b \,\middle|\, \begin{array}{l} \mathbf{ot}_1 \leftarrow \mathsf{OT}_1(1^\lambda, b; r) \\ \mathbf{ot}_2 \leftarrow \mathsf{OT}_2(1^\lambda, \ell_0, \ell_1, \mathbf{ot}_1) \end{array}\right] = 1.$$

- (T_R, ϵ_R)***-Computational Receiver Privacy.*** *For all machines $\mathcal{D}$ running in time at most $T_R(\lambda)$,*

$$\left|\Pr\left[\mathcal{D}(1^\lambda, \mathbf{ot}_{1,0}) = 1\right] - \Pr\left[\mathcal{D}(1^\lambda, \mathbf{ot}_{1,1}) = 1\right]\right| < \epsilon_R(\lambda),$$

  where $\mathbf{ot}_{1,b} \leftarrow \mathsf{OT}_1(1^\lambda, b)$ for $b \in \{0,1\}$, and the probability is taken over the coins of OT_1 and $\mathcal{D}$.
- ϵ_S***-Statistical Sender Privacy.*** *There exists a (possibly unbounded-time) extractor such that the following holds. For any sequence $\{\mathbf{ot}_{1,\lambda}, \ell_{0,\lambda}, \ell_{1,\lambda}\}_\lambda$, define the distribution ensembles $\{D_{0,\lambda}\}_\lambda$ and $\{D_{1,\lambda}\}_\lambda$, where $D_{b,\lambda}$ is defined as follows:*
  1. *Run $\mathsf{OT}_{\mathsf{extract}}(\mathbf{ot}_{1,\lambda})$ to obtain μ.*
  2. *If $b = 0$, output $\mathsf{OT}_2(1^\lambda, \ell_{0,\lambda}, \ell_{1,\lambda}, \mathbf{ot}_{1,\lambda})$.*
  3. *If $b = 1$, set $\ell'_b = \ell_{b,\lambda}$ and $\ell'_{1-b} = 0$ and output $\mathsf{OT}_2(1^\lambda, \ell'_0, \ell'_1, \mathbf{ot}_{1,\lambda})$.*

  The two ensembles $\{D_{0,\lambda}\}_\lambda$ and $\{D_{1,\lambda}\}_\lambda$ have statistical distance at most ϵ_S.

In the body of our paper, we will use the following syntax, which reduces trivially to the syntax above.

- $\mathsf{OT}_1(1^\lambda, x; r) \to \mathbf{ot}_1$: The receiver's OT_1 algorithm takes a string of choice bits x and produces the receiver's OT message.
- $\mathsf{OT}_2(1^\lambda, \mathsf{lab}, \mathbf{ot}_1; r') \to \mathbf{ot}_2$: The sender's OT_2 algorithm takes a list $\mathsf{lab} = \{\mathsf{lab}_{i,b}\}_{i \in [|x|], b \in \{0,1\}}$ of pairs of strings to choose from of length $|x|$ along with the receiver's OT message and produces the sender's OT message.
- $\mathsf{OT}_3(\mathbf{ot}_2; r)$: The receiver's OT_3 takes the sender's OT message and outputs $\{\mathsf{lab}_{i,x_i}\}_{i \in [|x|]}$.

2.3 Definition of Concurrent MPC

In this section, we present the definition of concurrent secure multi-party computation. The definition below is a generalization of the definition of concurrent secure multi/two-party computation [58, 64]. Parts of this section are taken verbatim from [64], where the main modifications are due to the fact that we allow the simulator to run in super-polynomial time.

Multi-party computation. Consider an n-party quantum functionality specified by a family of circuits $\mathcal{F} = \{\mathcal{F}_\lambda\}_\lambda$ where $\mathcal{F}_\lambda$ has $m_1(\lambda) + \cdots + m_n(\lambda)$ input bits and $\ell_1(\lambda) + \cdots + \ell_n(\lambda)$ output bits. Let Π be an n-party protocol for computing $\mathcal{F}$. For security parameter λ and any collection of inputs $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$, where $\mathbf{x}_i \in \{0,1\}^{m_i(\lambda)}$. We denote the output of the functionality by $\mathcal{F}_\lambda(\mathbf{x}_1, \ldots, \mathbf{x}_n) \to (\mathbf{y}_1, \ldots, \mathbf{y}_n)$, where $\mathbf{y}_i \in \{0,1\}^{\ell_i(\lambda)}$ and $\mathbf{x}_i$ is P_i's input.

Concurrent execution in the ideal model. Next, we describe the concurrent execution of the protocol in the ideal world. Unlike the stand-alone setting, here the trusted party computes the functionality many times, each time upon different inputs. Let $\Pi := (P_1, \ldots, P_n)$ be an MPC protocol for computing an n-ary circuit $\mathcal{F}$ and λ be the security parameter. We consider adversaries that corrupt any subset of the parties, where the subset is pre-determined before the beginning of the execution, and we denote by $I \subset [n]$ the subset of corrupted parties. An ideal execution with an adversary who controls the parties I proceeds as follows:

- **Inputs**: The inputs of the parties $P_1, \ldots, P_n$ are determined by input-selecting machines $M := M_1, \ldots, M_n$, where each M_i sends $\mathbf{x}_{i,j}$ to each party P_i and for each session j, at the beginning of the experiment.
- **Session initiation:** When the adversary initiates the session number j by sending a $(\textit{start-session}, i)$ to the trusted party. If $i \in [n] - I$ (means P_i is an honest party) the trusted party sends $(\textit{start-session}, j)$ to P_i, where $i \in [n]$, and j is the index of the session (i.e., this is the j-th session to be started by P_i).
- **Honest parties send inputs to trusted party:** Upon receiving the activation message $(\textit{start-session}, i)$ from the trusted party, each honest party P_i sends $(j, \mathbf{x}_{i,j})$ to the trusted party.
- **Corrupted parties send inputs to trusted party:** Whenever the adversary wishes, it may ask a corrupted party P_i to send a message $(j, \mathbf{x}'_{i,j})$ to the trusted third party, for any $\mathbf{x}'_{i,j}$ of its choice. A corrupted party P_i can send the pairs $(j, \mathbf{x}'_{i,j})$ in any order it wishes. The only limitation is that for any j, at most one pair indexed by j can be sent to the trusted party.
- **Trusted party answers corrupted parties:** When the trusted third party has received messages $(j, \mathbf{x}'_{i,j})$ from all parties (both honest and corrupted) it computes $\mathcal{F}(\mathbf{x}'_{1,j}, \ldots, \mathbf{x}'_{n,j}) \to (\mathbf{y}_{1,j}, \ldots, \mathbf{y}_{n,j})$ and sends $(j, \mathbf{y}_{i,j})$ to every corrupted P_i.
- **Adversary instructs the trusted party to answer honest parties:** When the adversary sends a message of the type $(\textit{send-output}, j, i)$ to the

trusted party, the trusted party directly sends $(j, \mathbf{y}_{i,j})$ to the honest party P_i. If all inputs for session j have not yet been received by the trusted party the message is ignored. If the output has already been delivered to the honest party, or i is the index so that P_i is a corrupted party, the message is ignored as well.

– **Outputs:** Each honest party always outputs the vector of outputs that it received from the trusted party. The corrupted parties may output an arbitrary state and the messages obtained from the trusted party.

Let $\mathcal{S}$ be a PPT algorithm (representing the ideal-model adversary) and let $I \subset [n]$ be the set of corrupted parties. The adversary takes as an input an auxiliary information $\mathbf{z}$. Then the ideal execution of $\mathcal{F}$, denoted by the random variable

$$IDEAL_{\mathcal{F},I,\mathcal{S},M}(\lambda, \mathbf{z})$$

is defined as the outputs of the ideal functionality and the output of $\mathcal{S}$, from the ideal process described above.

Execution in the real model. We next consider the execution of Π in the real world. We assume that the parties communicate through an asynchronous fully connected and authentic point-to-point channel but without guaranteed delivery of messages. Let $\mathcal{F}$, I be as above, and let Π be a multi-party protocol for computing the corresponding circuit. Furthermore, let $\mathcal{A}$ be a PPT machine such that for every $i \in I$, the adversary $\mathcal{A}$ controls P_i. Then, the real concurrent execution of Π with security parameter λ, and auxiliary input $\mathbf{z}$ to $\mathcal{A}$, is denoted

$$REAL_{\Pi,I,\mathcal{A}}(\lambda, \mathbf{z})$$

and it is defined as the output vector of the honest parties and the adversary $\mathcal{A}$ resulting from the following process. The parties run concurrent executions of the protocol, where every party initiates a new session whenever it receives a start-session from the adversary. The honest parties use the string provided by the attacker as their input for this session. The scheduling of all messages throughout the executions is controlled by the adversary.

Security. The security of Π under composition is defined by saying that for every real-model adversary there exists an ideal model adversary that can simulate the execution of the secure real-model protocol. We parametrize the definition by the runtime of the simulator T. Formally:

Definition 3 (Concurrent Security in the Malicious Model). *let $\mathcal{F}, n, \lambda$ and Π be as above. Protocol Π is said to T-securely realize $\mathcal{F}$ under concurrent composition if for every real-model PPT adversary $\mathcal{A}$, there exists an ideal-model adversary $\mathcal{S}$ with runtime bounded by T, such that every state $\mathbf{z}$ and every $I \subset [n]$ it holds that*

$$\{IDEAL_{\mathcal{F},I,\mathcal{S}}(\lambda, \mathbf{z})\}_{n \in \mathrm{N}} \approx_c \{REAL_{\Pi,I,\mathcal{A}}(\lambda, \mathbf{z})\}_{n \in \mathrm{N}},$$

where the notation $\approx_c$ denotes computational indistinguishability.

Remark 1. We shall pointed out that the above definition can be also generalized to real-world adversaries $\mathcal{A}$ beyond polynomial-time. Furthermore, our proof in Section 3.3 can be adapted to establish security for a real-world adversary whose runtime is bounded by $\tilde{T}$, and the simulator is bounded by some $T \gg \tilde{T}$.

Quantum Circuits. We can modify our definition to work with quantum circuits, how ever we keep the inputs and outputs classical. In particular, we only allow inputs $\boldsymbol{x}_i \in \{0,1\}^{m_i(\lambda)}$ and outputs $\boldsymbol{y}_i \in \{0,1\}^{\ell_i(\lambda)}$. The only differences are:

1. $\mathcal{F}$ is a family of quantum circuits.
2. $\mathcal{A}$ is a quantum circuit with auxiliary (quantum) state $\boldsymbol{z}$.
3. $\mathcal{S}$ runs in quantum polynomial time.

We leave it as an open problem to study the definition of quantum concurrent multiparty computation with inputs and outputs as quantum states.

Remark 2. We remark that our definition assumes that the honest parties' outputs are only revealed to the distinguisher at the end of the experiment and that it cannot adaptively choose honest parties' inputs in subsequent sessions based on previous honest outputs. Such a definition suffices for many applications concurrent 2PC, although we remark that stronger variants exist [27].

3 The Construction

In this section, we prove the following theorem.

Theorem 2. *Assuming the existence of subexponentially-secure versions of the following primitives:*

- *A two-round SPSS zero-knowledge argument system*
- *A two-message concurrent NMC scheme*
- *A two-round statistically-sender private oblivious transfer scheme*
- *A garbled circuit scheme*

there exists a two-round two-party computation protocol for any polynomial-time functionality f, in the plain model with super-polynomial simulation.

We note that each primitive is known from the subexponential hardness of LWE. In particular, [19] show the existence of two-round Statistically-Sender-Private OT from LWE, and both the SPSS zero-knowledge argument and the NMC scheme of [51] can be instantiated using LWE (see Section 4 for details). Finally, garbled circuits can be instantiated using any one-way function, which is known from LWE. Thus we have Theorem 2 as a corollary.

We now describe the construction of two-round two-party computation where both parties receive outputs.

$(\mathbf{nmc}_1^{\mathsf{P}_i,\mathsf{send}}, \mathbf{nmc}_1^{\mathsf{P}_j,\mathsf{recv}}, \mathbf{nmc}_1^{\mathsf{P}_j,\mathsf{send}}, \mathbf{nmc}_2^{\mathsf{P}_i,\mathsf{send}}, \tilde{C}_i^{\mathsf{P}}, \mathbf{ot}_1^{\mathsf{P}_j}, \mathbf{ot}_2^{\mathsf{P}_i}) \in L_{i\to j}$ **iff:**

There exists $(\mathbf{x}_i, r_c^{\mathsf{P}_i,\mathsf{send}}, r_{gc}^{\mathsf{P}_i}, r_{\mathbf{ot}}^{\mathsf{P}_i,\mathsf{send}})$ where

- $\mathbf{nmc}_1^{\mathsf{P}_i,\mathsf{send}} = \mathsf{NMC}_1^{\mathsf{send}}(1^\lambda, val; r_c^{\mathsf{P}_i,\mathsf{send}})$ for the value $val = (\mathbf{x}_i, r_{gc}^{\mathsf{P}_i}, r_{\mathbf{ot}}^{\mathsf{P}_i,\mathsf{send}})$,
- $\mathbf{nmc}_2^{\mathsf{P}_i,\mathsf{send}} = \mathsf{NMC}_2^{\mathsf{send}}(1^\lambda, val, \mathbf{nmc}_1^{\mathsf{P}_j,\mathsf{recv}}, r_c^{\mathsf{P}_i,\mathsf{send}})$,
- $(\tilde{C}_i^{\mathsf{P}}, \mathsf{lab}) = \mathsf{Garble}(C, r_{gc}^{\mathsf{P}_i})$ for the circuit C defined below, with the hardcoded value set to $(\mathbf{x}_i, \mathbf{nmc}_1^{\mathsf{P}_j,\mathsf{send}})$, and
- $\mathbf{ot}_2^{\mathsf{P}_i} = \mathsf{OT}_2(\mathsf{lab}, \mathbf{ot}_1^{\mathsf{P}_j}, r_{\mathbf{ot}}^{\mathsf{P}_i,\mathsf{send}})$, where lab is the family of labels obtained from Garble.

Fig. 1. Description of the language $\mathcal{L}_{i\to j}$

3.1 Required Primitives

First, we review the syntax of all the primitives we will use.

Let λ be the security parameter, and we assume 1^λ is an implicit parameter in all the following algorithms.

- A two-round $(T_{\mathsf{sound}}, T_{\mathsf{Sim}}, T_{\mathsf{zk}}, T_{\mathcal{L}}, \epsilon_1, \epsilon_2)$-SPSS ZK argument system

$$(\mathsf{ZK}_1, \mathsf{ZK}_2, \mathsf{ZK}_{\mathsf{verify}}, \mathsf{ZK}_{\mathsf{sim}}),$$

  where $T_{\mathsf{sound}}, T_{\mathsf{Sim}}, T_{\mathsf{zk}}, T_{\mathcal{L}}$ are specified below and ϵ_1, ϵ_2 are any negligible functions.
- A two-round $(T_{\mathbf{nmc}}, \epsilon)$-fully-concurrent non-malleable commitment scheme

$$(\mathsf{NMC}_1^{\mathsf{send}}, \mathsf{NMC}_1^{\mathsf{recv}}, \mathsf{NMC}_2^{\mathsf{send}}),$$

  where $T_{\mathbf{nmc}}$ is specified below and ϵ is any negligible function. In addition, we assume that the extraction algorithm $\mathsf{NMC}_{\mathsf{extract}}$ runs in time $T_{\mathsf{NMC}_{\mathsf{extract}}}$.
- A (T_G, ϵ)-garbled circuit scheme $(\mathsf{Garble}, \mathsf{Eval}, \mathsf{Sim}_{\mathsf{Garble}})$, where T_G is specified below and ϵ is any negligible function.
- A two-round $(T_R, \epsilon_1, \epsilon_2)$-statistically-sender-private OT scheme $(\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3, \mathsf{OT}_{\mathsf{extract}})$, where T_R is specified below and ϵ_1, ϵ_2 are any negligible functions. Additionally, we assume the extraction algorithm $\mathsf{OT}_{\mathsf{extract}}$ runs in time $T_{\mathsf{OT}_{\mathsf{extract}}}$.

For the zero-knowledge system, we define a language $L_{i\to j}$ in NP which will be proved by both parties during the 2PC protocol.

Complexity Hierarchy. We require the primitives above satisfy the following complexity hierarchy:

$$\mathsf{poly}(\lambda) \ll T_{\mathsf{sound}} \ll T_{\mathsf{Sim}} \ll T_{\mathbf{nmc}} \ll T_{\mathsf{NMC}_{\mathsf{extract}}} \ll T_{\mathsf{zk}} \ll T_R \ll T_{\mathsf{OT}_{\mathsf{extract}}} \ll T_G \ll T_L.$$

Additionally, we require that the language above is decidable in time T_L.

Some Final Notation. Let $\mathsf{onlychoices}(x, \mathsf{lab})$ take a string x and a list $\mathsf{lab} = \{\mathsf{lab}_{i,b}\}_{i\in[|x|],b\in\{0,1\}}$ of strings as input and produce the list $\{\mathsf{lab}'_{i,b}\}_{i\in[|x|],b\in\{0,1\}}$, where for each i $\mathsf{lab}'_{i,x_i} = \mathsf{lab}_{i,x_i}$, and $\mathsf{lab}'_{i,1-x_i} = 0$.

3.2 The Protocol

We now describe the protocol for two-round 2PC. Without loss of generality, we describe the actions of Party 1.

2-round 2PC protocol:

In each round, Party 1 performs the following actions.

Round 1:

1. Choose random strings $r_c^{\mathsf{P}_1,\mathsf{send}}, r_c^{\mathsf{P}_1,\mathsf{recv}}, r_{gc}^{\mathsf{P}_1}, r_{\mathbf{ot}}^{\mathsf{P}_1,\mathsf{recv}}, r_{\mathbf{ot}}^{\mathsf{P}_1,\mathsf{send}}$, and $r_{zk}^{\mathsf{P}_1}$ of appropriate sizes.
2. Compute a ZK verifier's message $\mathsf{zk}_1^{\mathsf{P}_1} \leftarrow \mathsf{ZK}_1(1^\lambda; r_{zk}^{\mathsf{P}_1})$.
3. Compute a round-one committer's NMC message
$$\mathbf{nmc}_1^{\mathsf{P}_1,\mathsf{send}} \leftarrow \mathsf{NMC}_1^{\mathsf{send}}(1^\lambda, val; r_c^{\mathsf{P}_1,\mathsf{send}}),$$
where the committed value $val = (\mathbf{x}_1, r_{gc}^{\mathsf{P}_1}, r_{\mathbf{ot}}^{\mathsf{P}_1,\mathsf{send}})$ consists of P_1's input along with the randomness which P_1 will use to generate the garbled circuit and OT2 messages in round 2.
4. Compute a round-one receiver's NMC message
$$\mathbf{nmc}_1^{\mathsf{P}_1,\mathsf{recv}} \leftarrow \mathsf{NMC}_1^{\mathsf{recv}}(1^\lambda; r_c^{\mathsf{P}_1,\mathsf{recv}}).$$
5. Compute an OT receiver's message
$$\mathbf{ot}_1^{\mathsf{P}_1} \leftarrow \mathsf{OT}_1(1^\lambda, (\mathbf{x}_1, r_c^{\mathsf{P}_1,\mathsf{send}}, r_{gc}^{\mathsf{P}_1}, r_{\mathbf{ot}}^{\mathsf{P}_1,\mathsf{send}}); r_{\mathbf{ot}}^{\mathsf{P}_1,\mathsf{recv}}),$$
where the choice bits $(\mathbf{x}_1, r_c^{\mathsf{P}_1,\mathsf{send}}, r_{gc}^{\mathsf{P}_1}, r_{\mathbf{ot}}^{\mathsf{P}_1,\mathsf{send}})$ consist of the randomness $r_c^{\mathsf{P}_1,\mathsf{send}}$ used to generate the round-one sender's NMC message along with the committed values $\mathbf{x}_1, r_{gc}^{\mathsf{P}_1}, r_{\mathbf{ot}}^{\mathsf{P}_1,\mathsf{send}}$.
6. Send $(\mathsf{zk}_1^{\mathsf{P}_1}, \mathbf{nmc}_1^{\mathsf{P}_1,\mathsf{send}}, \mathbf{nmc}_1^{\mathsf{P}_1,\mathsf{recv}}, \mathbf{ot}_1^{\mathsf{P}_1})$ to P_1.

Round 2:

After receiving the first-round message ($\mathsf{zk}_1^{\mathsf{P}_2}$, $\mathbf{nmc}_1^{\mathsf{P}_2,\mathsf{send}}$, $\mathbf{nmc}_1^{\mathsf{P}_2,\mathsf{recv}}$, $\mathbf{ot}_1^{\mathsf{P}_2}$) from party 2, party 1 does the following:

1. Compute the sender's second-round NMC message

$$\mathbf{nmc}_2^{\mathsf{P}_1,\mathsf{send}} \leftarrow \mathsf{NMC}_2^{\mathsf{send}}(1^\lambda, val, \mathbf{nmc}_1^{\mathsf{P}_2,\mathsf{recv}}, r_c^{\mathsf{P}_1,\mathsf{send}}),$$

   where the committed value $val = (\mathbf{x}_1, r_{gc}^{\mathsf{P}_1}, r_{\mathbf{ot}}^{\mathsf{P}_1,\mathsf{send}})$ is as in round 1.
2. Compute the garbled circuit $(\tilde{C}^{\mathsf{P}_1}, \mathsf{lab}) \leftarrow \mathsf{Garble}(1^\lambda, C, r_{gc}^{\mathsf{P}_1})$, where C is the circuit defined below, and the hardcoded values are $(\lambda, \mathbf{x}_1, \mathbf{nmc}_1^{\mathsf{P}_2,\mathsf{send}})$.
3. Compute the sender's OT message

$$\mathbf{ot}_2^{\mathsf{P}_1} \leftarrow \mathsf{OT}_2(1^\lambda, \mathsf{lab}, \mathbf{ot}_1^{\mathsf{P}_2}; r_{\mathbf{ot}}^{\mathsf{P}_1,\mathsf{send}}),$$

   with the labels lab obtained from the garbling algorithm in the previous step.
4. Compute the prover's ZK message $\mathsf{zk}_2^{\mathsf{P}_1} \leftarrow \mathsf{ZK}_2(1^\lambda, \phi, w, \mathsf{zk}_1^{\mathsf{P}_2})$ for the language $L_{1\to 2}$ with the statement

$$\phi = (\mathbf{nmc}_1^{\mathsf{P}_1,\mathsf{send}}, \mathbf{nmc}_1^{\mathsf{P}_2,\mathsf{recv}}\mathbf{nmc}_1^{\mathsf{P}_2,\mathsf{send}}, \mathbf{nmc}_2^{\mathsf{P}_1,\mathsf{send}}, \tilde{C}^{\mathsf{P}_1}, \mathbf{ot}_1^{\mathsf{P}_2}, \mathbf{ot}_2^{\mathsf{P}_1})$$

   and witness $w = (\mathbf{x}_1, r_c^{\mathsf{P}_1,\mathsf{send}}, r_{gc}^{\mathsf{P}_1}, r_{\mathbf{ot}}^{\mathsf{P}_1,\mathsf{send}})$.
5. Send $(\mathbf{nmc}_2^{\mathsf{P}_1,\mathsf{send}}, \tilde{C}^{\mathsf{P}_1}, \mathbf{ot}_2^{\mathsf{P}_1}, \mathsf{zk}_2^{\mathsf{P}_1})$ to P_2.

Output Computation:

After receiving party 2's second-round message $(\mathbf{nmc}_2^{\mathsf{P}_2,\mathsf{send}}, \tilde{C}^{\mathsf{P}_2}, \mathbf{ot}_2^{\mathsf{P}_2}, \mathsf{zk}_2^{\mathsf{P}_2})$, party 1 does the following to compute its output:

1. If the NMC verification algorithm $\mathsf{NMC}_{\mathsf{verify}}(1^\lambda, \tau, r_c^{\mathsf{P}_1,\mathsf{recv}})$ fails with respect to P_2's commitment transcript

$$\tau = (\mathbf{nmc}_1^{\mathsf{P}_2,\mathsf{send}}, \mathbf{nmc}_1^{\mathsf{P}_1,\mathsf{recv}}, \mathbf{nmc}_2^{\mathsf{P}_1,\mathsf{send}}),$$

   then abort and output $\perp$.
2. Let

$$\phi' = (\mathbf{nmc}_1^{\mathsf{P}_2,\mathsf{send}}, \mathbf{nmc}_1^{\mathsf{P}_1,\mathsf{recv}}\mathbf{nmc}_1^{\mathsf{P}_1,\mathsf{send}}, \mathbf{nmc}_2^{\mathsf{P}_2,\mathsf{send}}, \tilde{C}^{\mathsf{P}_2}, \mathbf{ot}_1^{\mathsf{P}_1}, \mathbf{ot}_2^{\mathsf{P}_2})$$

   be the statement which party 2 proves via $\mathsf{zk}_2^{\mathsf{P}_2}$, with respect to language $L_{2\to 1}$. If $\mathsf{ZK}_{\mathsf{verify}}(\phi', \mathsf{zk}_2^{\mathsf{P}_2}, r_{zk}^{\mathsf{P}_1}) = 0$ then abort and output $\perp$.
3. Compute the output labels $\mathsf{lab}' \leftarrow \mathsf{OT}_3(\mathbf{ot}_2^{\mathsf{P}_2}, r_{\mathbf{ot}}^{\mathsf{P}_1,\mathsf{recv}})$ of the OT protocol.
4. Output the evaluation $\mathsf{Eval}(\tilde{C}^{\mathsf{P}_2}, \mathsf{lab}')$ of the garbled circuit $\tilde{C}^{\mathsf{P}_2}$ sent by P_2, using the labels lab' obtained in the previous step.

The circuit C which is garbled by Party 1 is as follows.

Circuit C:

Input: $(\mathbf{x}_2, r_c^{\mathsf{P}_2,\mathsf{send}}, r_{gc}^{\mathsf{P}_2}, r_{\mathbf{ot}}^{\mathsf{P}_2,\mathsf{send}})$
Hardcoded: $(\lambda, \mathbf{x}_1, \mathbf{nmc}_1^{\mathsf{P}_2,\mathsf{send}})$

1. **If** $\mathbf{nmc}_1^{\mathsf{P}_2,\mathsf{send}} = \mathsf{NMC}_1^{\mathsf{send}}(1^\lambda, (\mathbf{x}_2, r_{gc}^{\mathsf{P}_2}, r_{\mathbf{ot}}^{\mathsf{P}_2,\mathsf{send}}); r_c^{\mathsf{P}_2,\mathsf{send}})$, **then:**
   (a) Return $f(\mathbf{x}_1, \mathbf{x}_2)$
2. **Else:**
   (a) Return $\perp$.

3.3 Security

We now prove Theorem 2 by showing that the protocol above satisfies the definition of concurrent MPC security given in Section 2.3.

Let there be n parties, with a subset of corrupted parties $\mathcal{C} \in [n]$. Consider a PPT adversary $\mathcal{A}$ which spawns a polynomial number of sessions of the protocol described above, where for each session at most one party is corrupt, and schedules messages across the different sessions in an arbitrary order, controlling the inputs and messages of the corrupted parties. At the end of the experiment, $\mathcal{A}$ receives the outputs of all parties in all sessions. We show the existence of an ideal-world adversary (called the "simulator") which produces an interaction with $\mathcal{A}$ that is indistinguishable from the real-world interaction of $\mathcal{A}$.

We describe the behavior of the simulator below. In the following, we denote a session by (s, i, j), where s is the session number, and parties P_i and P_j run the 2PC protocol during this session. Without loss of generality we assume P_i is honest and P_j is corrupt, and that $\mathcal{A}$ always asks for the message of P_i in both rounds before sending the the message of P_j for that round.

The Concurrent-Secure Simulator:

At the beginning of the experiment, the simulator invokes $\mathcal{A}$. The simulator also initializes a database where it will store, for each session (s, i, j), the messages and extracted values of P_j, the simulator's private state for this session, along with the ideal functionality output for the session. The simulator then responds to $\mathcal{A}$ in the following manner.

Whenever $\mathcal{A}$ initializes session (s, i, j), do the following to simulate P_i's message to P_j:

1. Choose random strings $r_c^{P_i,\mathsf{send}}, r_c^{P_i,\mathsf{recv}}, r_{gc}^{P_i}, r_{ot}^{P_i,\mathsf{recv}}, r_{ot}^{P_i,\mathsf{send}}$, and $r_{zk}^{P_i}$ of appropriate sizes. Store all strings as the simulator's private state for session (s, i, j).
2. Compute a ZK verifier's message $\mathsf{zk}_1^{P_i} \leftarrow \mathsf{ZK}_1(1^\lambda; r_{zk}^{P_i})$.

3. Compute a round-one committer's NMC message

$$\mathbf{nmc}_1^{P_i,\mathsf{send}} \leftarrow \mathsf{NMC}_1^{\mathsf{send}}(1^\lambda, val; r_c^{P_i,\mathsf{send}})$$

   for the value $val = (0,0,0)$.
4. Compute a round-one receiver's NMC message

$$\mathbf{nmc}_1^{P_i,\mathsf{recv}} \leftarrow \mathsf{NMC}_1^{\mathsf{recv}}(1^\lambda; r_c^{P_i,\mathsf{recv}}).$$

5. Compute an OT receiver's message $\mathbf{ot}_1^{P_i} \leftarrow \mathsf{OT}_1(1^\lambda, (0,0,0,0); r_{ot}^{P_i,\mathsf{recv}})$, where the choice bits are $(0,0,0,0)$.
6. Send $(\mathsf{zk}_1^{P_i}, \mathbf{nmc}_1^{P_i,\mathsf{send}}, \mathbf{nmc}_1^{P_i,\mathsf{recv}}, \mathbf{ot}_1^{P_i})$ to P_j on behalf of P_i.

Whenever $\mathcal{A}$ sends a first-round message m on behalf of P_j in session (s,i,j), do the following:

1. Parse m as $(\mathsf{zk}_1^{P_j}, \mathbf{nmc}_1^{P_j,\mathsf{send}}, \mathbf{nmc}_1^{P_j,\mathsf{recv}}, \mathbf{ot}_1^{P_j})$. Store m as P_j's first-round message in session (s,i,j).
2. Compute the extracted values $(\mathbf{x}_j, r_c^{P_j,\mathsf{send}}, r_{gc}^{P_j}, r_{ot}^{P_j,\mathsf{send}}) \leftarrow \mathsf{OT}_{\mathsf{extract}}(\mathbf{ot}_1^{P_j})$ from P_j's OT receiver's message, and save $(\mathbf{x}_j, r_c^{P_j,\mathsf{send}}, r_{gc}^{P_j}, r_{ot}^{P_j,\mathsf{send}})$ as P_j's OT receiver value in session (s,i,j).
3. If $\mathbf{nmc}_1^{P_j,\mathsf{send}} = \mathsf{NMC}_1^{\mathsf{send}}(1^\lambda, (\mathbf{x}_j, r_{gc}^{P_j}, r_{ot}^{P_j,\mathsf{send}}); r_c^{P_j,\mathsf{send}})$, then send $\mathbf{x}_j$ to the ideal functionality and receive back the evaluation $y = f(\mathbf{x}_i, \mathbf{x}_j)$. If $\mathbf{nmc}_1^{P_j,\mathsf{send}} \neq \mathsf{NMC}_1^{\mathsf{send}}(1^\lambda, (\mathbf{x}_j, r_{gc}^{P_j}, r_{ot}^{P_j,\mathsf{send}}); r_c^{P_j,\mathsf{send}})$, set $y = \perp$.
4. Store y as the ideal-world output for P_j in session (s,i,j).

Whenever $\mathcal{A}$ requests a second-round message from honest party P_i in session (s,i,j), do the following:

1. Retrieve P_j's first-round message $m = (\mathsf{zk}_1^{P_j}, \mathbf{nmc}_1^{P_j,\mathsf{send}}, \mathbf{nmc}_1^{P_j,\mathsf{recv}}, \mathbf{ot}_1^{P_j})$ for session (s,i,j).
2. Compute a round-two NMC sender's message

$$\mathbf{nmc}_2^{P_i,\mathsf{send}} \leftarrow \mathsf{NMC}_2^{\mathsf{send}}(1^\lambda, val, \mathbf{nmc}_1^{P_j,\mathsf{recv}}, r_c^{P_i,\mathsf{send}})$$

   for the value $val = (0,0,0)$.
3. Compute a simulated garbled circuit $(\tilde{C}_i^P, \mathsf{lab}) \leftarrow \mathsf{Sim}_{\mathsf{Garble}}(1^\lambda, |C|, y; r_{gc}^{P_i})$ using the output y saved previously for session (s,i,j).
4. Compute an OT sender's message $\mathbf{ot}_2^{P_i} \leftarrow \mathsf{OT}_2(1^\lambda, \mathsf{onlychoices}(c, \mathsf{lab}), \mathbf{ot}_1^{P_j}, r_{ot}^{P_i,\mathsf{send}})$, where $c = (\mathbf{x}_j, r_c^{P_j,\mathsf{send}}, r_{gc}^{P_j}, r_{ot}^{P_j,\mathsf{send}})$ is the saved OT receiver's value for round (s,i,j). Recall that $\mathsf{onlychoices}$ sets all non-chosen labels to 0.
5. Compute a simulated prover's ZK message $\mathsf{zk}_2^{P_i} \leftarrow \mathsf{ZK}_{\mathsf{sim}}(1^\lambda, \phi, \mathsf{zk}_1^{PTwo}, r')$ using the statement

$$\phi = (\mathbf{nmc}_1^{\mathsf{P}_1,\mathsf{send}}, \mathbf{nmc}_1^{\mathsf{P}_2,\mathsf{recv}} \mathbf{nmc}_1^{\mathsf{P}_2,\mathsf{send}}, \mathbf{nmc}_2^{\mathsf{P}_1,\mathsf{send}}, \tilde{C}^{\mathsf{P}_1}, \mathbf{ot}_1^{\mathsf{P}_2}, \mathbf{ot}_2^{\mathsf{P}_1})$$

   and r' is random.
6. Send $(\mathbf{nmc}_2^{P_i,\mathsf{send}}, \tilde{C}_i^P, \mathbf{ot}_2^{P_i}, \mathsf{zk}_2^{P_i})$ to P_j on behalf of P_i.

Whenever $\mathcal{A}$ sends a second-round message m on behalf of P_j for session (s, i, j), do the following:

1. Parse m as $(\mathbf{nmc}_2^{P_j,\mathsf{send}}, \tilde{C}_j^P, \mathbf{ot}_2^{P_j}, \mathsf{zk}_2^{P_j})$.
2. If the NMC verification algorithm $\mathsf{NMC}_{\mathsf{verify}}(1^\lambda, \tau, r_c^{P_i,\mathsf{recv}})$ fails with respect to P_j's commitment transcript $\tau = (\mathbf{nmc}_1^{P_j,\mathsf{send}}, \mathbf{nmc}_1^{P_i,\mathsf{recv}}, \mathbf{nmc}_2^{P_i,\mathsf{send}})$, then instruct the ideal functionality to deliver $\perp$ to P_i.
3. If $\mathsf{ZK}_{\mathsf{verify}}(\phi', \mathsf{zk}_2^{P_i}, r_{zk}^{P_j}) = 0$ then instruct the ideal functionality to deliver $\perp$ to P_i.
4. Extract the committed values

$$(\mathbf{x}_j, r_{gc}^{P_j}, r_{ot}^{P_j,\mathsf{send}}) \leftarrow \mathsf{NMC}_{\mathsf{extract}}(\mathbf{nmc}_1^{P_j,\mathsf{send}}, \mathbf{nmc}_1^{P_i,\mathsf{recv}}, \mathbf{nmc}_2^{P_j,\mathsf{send}})$$

   from P_j's NMC transcript. If we haven't already queried the ideal functionality, send $\mathbf{x}_j$ to the ideal functionality. Note that because the NMC is perfectly binding after round 1, the value $\mathbf{x}_j$ is identical to the value extracted by $\mathsf{OT}_{\mathsf{extract}}$ during round 2 as long as the identity checked in C holds.
5. Use the values obtained in the previous step to check if the conditions in statement ϕ hold with respect to language $L_{j\to i}$. If they do not hold, output "special abort".
6. If we have not yet aborted, instruct the ideal functionality to deliver the output to P_i.

We show the view in the real world is indistinguishable from the view in the ideal world via a series of hybrid games, where the first hybrid $\mathcal{H}_0$ corresponds to the real world and the last hybrid $\mathcal{H}_6$ corresponds to the ideal world. The hybrids are as follows.

Hybrid $\mathcal{H}_0$: In this hybrid, the simulator plays the role of all honest parties in all sessions, and behaves identically to the real-world executions of the protocol.

Hybrid $\mathcal{H}_1$: Here the simulator acts in the same way as in $\mathcal{H}_0$ except that for each honest party P_i's round 2 message during session (s, i, j) it simulates the ZK proof it sends to $\mathcal{A}$. This hybrid now runs in time $\mathsf{poly}(T_{\mathsf{Sim}})$.

Hybrid $\mathcal{H}_2$: The simulator acts in the same way as $\mathcal{H}_1$, except that when computing each honest party P_i's round 1 message during session (s, i, j), it sends the chooser's OT message with choice bits $(0, 0, 0, 0)$ instead of $(\mathbf{x}_i, r_c^{P_i,\mathsf{send}}, r_{gc}^{P_i}, r_{ot}^{P_i,\mathsf{send}})$. This hybrid still runs in time $\mathsf{poly}(T_{\mathsf{Sim}})$.

Hybrid $\mathcal{H}_3$: The simulator acts in the same way as $\mathcal{H}_2$, except for each session (s, i, j), during rounds 1 and 2 it commits to $(0, 0, 0)$ on behalf of P_i instead of $(\mathbf{x}_i, r_{gc}^{P_i}, r_{ot}^{P_i,\mathsf{send}})$. This hybrid still runs in time $\mathsf{poly}(T_{\mathsf{Sim}})$.

Hybrid $\mathcal{H}_4$: The simulator acts in the same way as $\mathcal{H}_3$, except that after receiving P_j's round 2 message during session (s, i, j) it breaks $\mathbf{nmc}_1^{P_j,\mathsf{send}}$ to obtain $(\mathbf{x}_j, r_{gc}^{P_j}, r_{ot}^{P_j,\mathsf{send}})$ and during the output computation phase outputs "special abort" if the conditions in statement ϕ don't hold. This hybrid now runs in time $\mathsf{poly}(T_{\mathsf{NMC}_{\mathsf{extract}}})$.

Hybrid $\mathcal{H}_5$: The simulator acts in the same way as $\mathcal{H}_4$, except that after receiving P_j's round 1 message during session (s, i, j), it runs $\mathsf{OT}_{\mathsf{extract}}$ on P_j's OT receiver message to obtain the values $(\mathbf{x}_j, r_c^{P_j,\mathsf{send}}, r_{gc}^{P_j}, r_{ot}^{P_j,\mathsf{send}})$. If $\mathbf{nmc}_1^{P_j,\mathsf{send}} = \mathsf{NMC}_1^{\mathsf{send}}(1^\lambda, (\mathbf{x}_j, r_{gc}^{P_j}, r_{ot}^{P_j,\mathsf{send}}); r_c^{P_j,\mathsf{send}})$, the simulator sends $\mathbf{x}_j$ to the ideal functionality to obtain $f(\mathbf{x}_i, \mathbf{x}_j)$. On the other hand, if $\mathbf{nmc}_1^{P_j,\mathsf{send}} \neq \mathsf{NMC}_1^{\mathsf{send}}(1^\lambda, (\mathbf{x}_j, r_{gc}^{P_j}, r_{ot}^{P_j,\mathsf{send}}); r_c^{P_j,\mathsf{send}})$, the simulator sends the value $\mathbf{x}_j$ extracted using $\mathsf{NMC}_{\mathsf{extract}}$ after receiving P_j's round 2 message to the ideal functionality. It tells the ideal functionality to deliver the output to P1 at the end of session (s, i, j) as long as the session did not abort. This hybrid now runs in time $\mathsf{poly}(T_{\mathsf{OT}_{\mathsf{extract}}})$.

Hybrid $\mathcal{H}_6$: The simulator acts in the same way as $\mathcal{H}_5$, except that for every honest party P_i's second-round message during session (s, i, j), it simulates the generation of the garbled circuit using the saved value y received from the ideal functionality instead of generating it honestly. This final hybrid runs in time $\mathsf{poly}(T_{\mathsf{OT}_{\mathsf{extract}}})$, which is the running time of the ideal-world simulator.

We want to use these hybrids to show the view of $\mathcal{A}$ is indistinguishable between the real and ideal worlds. There is a problem, though: in $\mathcal{H}_2$ and $\mathcal{H}_3$, the honest parties have no way to obtain its output. This is because the simulator switches the honest parties' $\mathbf{ot}_1$ messages to 0 in $\mathcal{H}_2$, which means the real-world method of running the garbled circuit to obtain the output will not work, and the simulator is not yet powerful enough to break the commitment.

Despite this, it is still possible to use this ordering of hybrids to prove indistinguishability. Consider the pair $(s, \mathbf{x}_j, b_i)$, where $\mathbf{x}_j$ is the input committed to by corrupt party P_j during session (s, i, j), and b_i is a bit which denotes whether or not honest party P_i accepts P_j's NMC and zero knowledge proof during the same session. Assuming $\mathcal{A}$ cannot generate a proof for a false statement, this pair determines the output of P_i in session (s, i, j) regardless of whether we are in the real or the ideal world. So to make the proof work, during certain steps we will argue indistinguishability of the tuple $(v, \{(s, \mathbf{x}_j, b_i)\}_s)$ between hybrids, where v is the view of $\mathcal{A}$.

The proof is organized as follows.

We first argue computational indistinguishability between each successive pair of hybrids. Afterwards we argue indistinguishablity of the combined view of $\mathcal{A}$ along with the output of P_1. Before starting, define a "bad" event $\mathcal{E}$ which will be useful in our proofs.

Definition 4. *We define event $\mathcal{E}$ to occur if there exists a session (s, i, j) where both of the following happen:*

1. *P_i accepts P_j's ZK proof*
2. *one of the conditions of the statement ϕ' do not hold w.r.t. $L_{j\to i}$.*

Lemma 1. *$\mathcal{E}$ occurs with negligible probability in $\mathcal{H}_0$.*

Proof. This follows from the adaptive soundness of the SPSS ZK argument system for languages decidable in time T_L and the fact that $L_{j\to i}$ is decidable in time T_L. □

Lemma 2. *Assuming the zero-knowledge property of the SPSS ZK argument system, the view of $\mathcal{A}$ between $\mathcal{H}_0$ and $\mathcal{H}_1$ are computationally indistinguishable.*

Proof. We prove the claim via a sequence of subhybrids for each session (s, i, j), where in each subhybrid we switch to a simulated ZK2 message for P_i.

Assume there is a PPT adversary $\mathcal{A}$ who can interact with the simulator and then, given P_1's output, distinguish between real and simulated for session (s, i, j). Then we construct a PPT adversary $\mathcal{A}'$ which contradicts the zero-knowledge property of the ZK system.

Fix the randomness used by the adversary, and by the simulator to generate all honest parties' messages before P_i's second-round message. There must be at least one way to fix this randomness such that the advantage of $\mathcal{A}$ is still nonnegligible. This also fixes the statement ϕ (and the witness w for s) which P_i should prove in round 2.

Now we construct $\mathcal{A}'$ to run the experiment with this fixed randomness, and to forward $\mathsf{zk}_1^{P_j}$ to the ZK challenger. Then $\mathcal{A}'$ receives $\mathsf{zk}_2^{P_i}$ which is either a valid proof of s or a simulated one. $\mathcal{A}'$ uses $\mathsf{zk}_2^{P_i}$ as the proof to send to $\mathcal{A}$ instead of generating one itself when generating the second-round message for P_i. It then outputs whatever $\mathcal{A}$ outputs.

$\mathcal{A}$ distinguishes the real and simulated for (s, i, j) even with the round 1 randomness fixed, and this is identical to the experiment described above with the new $\mathcal{A}'$. So $\mathcal{A}'$ is a distinguisher for the zero-knowledge property of the ZK system. □

Lemma 3. *Assuming the zero-knowledge property of the SPSS ZK argument system, $\mathcal{E}$ occurs with negligible probability in $\mathcal{H}_1$.*

Proof. Assume there is an adversary $\mathcal{A}$ which causes $\mathcal{E}$ to happen with nonnegligible probability in $\mathcal{H}_1$. Note that by Lemma 1 $\mathcal{A}$ cannot cause $\mathcal{E}$ to happen with nonnegligible probability in $\mathcal{H}_0$. We can extract the committed value in time $T_{\mathsf{NMC}_{\mathsf{extract}}}$ for each session to check whether or not $\mathcal{E}$ holds, thus creating a $\mathsf{poly}(T_{\mathsf{NMC}_{\mathsf{extract}}})$-time distinguisher for $\mathcal{H}_0$ and $\mathcal{H}_1$, contradicting Lemma 2, since $\mathsf{poly}(T_{\mathsf{NMC}_{\mathsf{extract}}}) \ll T_{\mathsf{zk}}$. □

Lemma 4. *Assuming the chooser's security of the OT scheme, the tuple $(v, \{(s, \mathbf{x}_j, b_i)\}_s)$ between $\mathcal{H}_1$ and $\mathcal{H}_2$ is computationally indistinguishable.*

Proof. We prove the claim via a sequence of subhybrids for each session (s, i, j), where in each subhybrid we switch P_i's $\mathbf{ot}_1$ message to 0.

Assume there is a PPT adversary $\mathcal{A}$ who can interact with the simulator and then, given $\{(s, \mathbf{x}_j, b_i)\}_s$ in addition to its view v at the end of the interaction, distinguishes between the subhybrid for some (s, i, j) and the previous subhybrid with nonnegligible probability. We use $\mathcal{A}$ to build an adversary $\mathcal{A}'$ for the OT chooser's security game. For simplicity of exposition, we first assume that $\mathcal{A}$ distinguishes only given its view v. Once we have established the reduction in this case, we extend it to the case where $\mathcal{A}$ also receives $(v, \{(s, \mathbf{x}_j, b_i)\}_s)$.

Fix the randomness $(r_c^{P_i,\mathsf{send}}, r_{gc}^{P_i}, r_{ot}^{P_i,\mathsf{send}})$ generated on behalf of P_i for session (s,i,j). There must be at least one such fixed value for which $\mathcal{A}$ still distinguishes with nonnegligible probability. Let $\mathcal{A}'$ run the experiment identically to the previous subhybrid with the randomness above fixed to this particular value, except that instead of computing $\mathbf{ot}_1^{P_i}$ directly it receives this value from the OT challenger. The challenger either computes the OT with choice bits $(r_c^{P_i,\mathsf{send}}, r_{gc}^{P_i}, r_{ot}^{P_i,\mathsf{send}})$ or $(0,0,0,0)$.

Assuming $(r_c^{P_i,\mathsf{send}}, r_{gc}^{P_i}, r_{ot}^{P_i,\mathsf{send}})$ is fixed, this experiment is identical to the previous subhybrid in the first case and the subhybrid for (s,i,j) in the second case. So if $\mathcal{A}$ successfully distinguishes then $\mathcal{A}'$ does as well. This contradicts chooser's security of the OT, since $T_{\mathsf{Sim}} \ll T_R$.

To extend to the case where $\mathcal{A}$ also receives $\{(s, \mathbf{x}_j, b_i)\}_s$, note that we can break the commitments of each of the corrupted parties in time $T_{\mathsf{NMC}_{\mathsf{extract}}}$ to retrieve each corrupted input $\mathbf{x}_j$, and b_i is known already by the experiment. Passing these to the adversary we obtain a $\mathsf{poly}(T_{\mathsf{NMC}_{\mathsf{extract}}})$-time distinguisher, which still contradicts chooser's security of the OT, since $\mathsf{poly}(T_{\mathsf{NMC}_{\mathsf{extract}}}) \ll T_R$. □

Lemma 5. *Assuming $\mathcal{E}$ occurs with negligible probability in $\mathcal{H}_1$, $\mathcal{E}$ occurs with negligible probability in $\mathcal{H}_2$.*

Proof. Assume there is an adversary $\mathcal{A}$ which causes $\mathcal{E}$ to happen with nonnegligible probability in $\mathcal{H}_2$. Note that by Lemma 3 $\mathcal{A}$ cannot cause $\mathcal{E}$ to happen with nonnegligible probability in $\mathcal{H}_1$. We can break the corrupted parties' commitments each in time $T_{\mathsf{NMC}_{\mathsf{extract}}}$ to create a $\mathsf{poly}(T_{\mathsf{NMC}_{\mathsf{extract}}})$-time distinguisher for $\mathcal{H}_1$ and $\mathcal{H}_2$, contradicting Lemma 4, since $\mathsf{poly}(T_{\mathsf{NMC}_{\mathsf{extract}}}) \ll T_R$. □

Lemma 6. *Assuming the non-malleability of the commitment scheme, the tuple $(v, \{(s, \mathbf{x}_j, b_i)\}_s)$ between $\mathcal{H}_2$ and $\mathcal{H}_3$ is computationally indistinguishable.*

Proof. We prove the claim via a sequence of subhybrids for each session (s,i,j), where, in each subhybrid we switch to an NMC of 0 for P_i.

Assume there is a PPT adversary $\mathcal{A}$ who can interact with the simulator and then, given $\{(s, \mathbf{x}_j, b_i)\}_s$ in addition to its view v at the end of the interaction, distinguishes between the previous hybrid and the hybrid for (s,i,j) with non-negligible advantage. Then we create a T_{Sim}-time $\mathcal{A}'$ which contradicts the non-malleability property of the commitment scheme. Note that $\{(s, \mathbf{x}_j, b_i)\}_s$ is computable directly from the output of the non-malleability game, since each corrupted party P_j commits to x_j (i.e., these are part of the RHS committed values).

Fix the randomness $r_{gc}^{P_i}$ and $r_{ot}^{P_i,\mathsf{send}}$ generated for P_i in session (s,i,j). There must be some such fixed values where $\mathcal{A}$ still distinguishes between the two subhybrids with non-negligible advantage. We create $\mathcal{A}'$ as follows. $\mathcal{A}'$ runs the experiment identically to the previous subhybrid, except that the values $r_{gc}^{P_i}$ and $r_{ot}^{P_i,\mathsf{send}}$ are fixed to maximize the probability of distinguishing, and the following changes are made to the non-malleable commitment interactions. When

computing P_i's round 1 message, instead of computing $\mathbf{nmc}_1^{P_i,\mathsf{send}}$, it receives this value from the challenger for the NMC game and forwards it to $\mathcal{A}$. It forwards $\mathbf{nmc}_1^{P_j,\mathsf{recv}}$ which it receives from $\mathcal{A}$ to the challenger as well. When computing P_i's round 2 message, it receives $\mathbf{nmc}_2^{P_i,\mathsf{send}}$ from the challenger and forwards it to $\mathcal{A}$ again. The NMC challenger commits to either $(\mathbf{x}_i, r_{gc}^{P_i}, r_{ot}^{P_i,\mathsf{send}})$ or $(0,0,0)$.

If the challenger commits to $(\mathbf{x}_i, r_{gc}^{P_i}, r_{ot}^{P_i,\mathsf{send}})$ then the experiment is identical to the subhybrid directly preceding the subhybrid for session (s,i,j), and if the challenger commits to $(0,0,0)$ then the experiment is identical to the subhybrid for (s,i,j) (with some fixed randomness, as described above). Thus $\mathcal{A}'$ wins the non-malleability game of the commitment scheme with non-negligible probability, contradicting the non-malleability of the commitment scheme, since $\mathsf{poly}(T_{\mathsf{Sim}}) \ll T_{\mathbf{nmc}}$. □

Lemma 7. *Assuming $\mathcal{E}$ occurs with negligible probability in $\mathcal{H}_2$ and the non-malleability of the commitment scheme, $\mathcal{E}$ occurs with negligible probability in $\mathcal{H}_3$.*

Proof. Assume that $\mathcal{E}$ occurs with non-negligible probability in $\mathcal{H}_3$. We can construct an adversary $\mathcal{A}'$ in the same way as in Lemma 6, playing the role of the adversary in a full nonmalleability game. By the nonmalleability property of the commitment scheme, the joint view of $\mathcal{A}'$ combined with the values it committed to are indistinguishable regardless of what the challenger commits to. If we have both the view of $\mathcal{A}'$ along with the values committed to it is easy to check if $\mathcal{E}$ occured. If $\mathcal{E}$ occurs with nonnegligible probability in $\mathcal{H}_2$ then by checking $\mathcal{E}$ we have a $\mathsf{poly}(T_{\mathsf{Sim}})$-time distinguisher which contradicts non-malleability of the NMC. □

Lemma 8. *Assuming $\mathcal{E}$ occurs with negligible probability in $\mathcal{H}_3$, then the tuple $(v, \{(s, \mathbf{x}_j, b_i)\}_s)$ between $\mathcal{H}_3$ and $\mathcal{H}_4$ are computationally indistinguishable.*

Proof. The only difference between $\mathcal{H}_3$ and $\mathcal{H}_4$ is that we break all corrupted parties' commitments and output "special abort" if at any point $\mathcal{E}$ occurred. So the only time the two hybrids are distinguishable is if $\mathcal{E}$ occurs. Thus indistinguishability follows from Lemma 7. □

Note that from this point onward, proving hybrid indistinguishability is sufficient for proving $\mathcal{E}$ occurs with negligible probability, since every hybrid now checks $\mathcal{E}$ explicitly.

Lemma 9. *The tuple $(v, \{(s, \mathbf{x}_j, b_i)\}_s)$ between $\mathcal{H}_4$ and $\mathcal{H}_5$ are computationally indistinguishable.*

Proof. This follows trivially from the fact that the view of $\mathcal{A}$ is identical between $\mathcal{H}_4$ and $\mathcal{H}_5$. □

Lemma 10. *Assuming $\mathcal{E}$ happens with negligible probability in $\mathcal{H}_1$ and $\mathcal{H}_5$, the view of the adversary $\mathcal{A}$ between $\mathcal{H}_1$ and $\mathcal{H}_5$ is computationally indistinguishable.*

Proof. By the previous claims the tuple $(v, \{(s, \mathbf{x}_j, b_i)\}_s)$ is indistinguishable between these hybrids. Assuming $\mathcal{E}$ did not happen, in both hybrids the output of each honest party P_i during session (s, i, j) is $f(\mathbf{x}_i, \mathbf{x}_j)$ if $b = 1$ and $\perp$ if $b = 0$. To see why this is the case when $b = 1$, note that the value $\mathbf{x}_j$ extracted by $\mathsf{OT}_{\mathsf{extract}}$ after round 1 is identical to the value extracted for $\mathbf{x}'_j$ by $\mathsf{NMC}_{\mathsf{extract}}$ after round 2. Assuming $\mathcal{E}$ does not occur, P1 outputs $\mathbf{x}'_j$ in $\mathcal{H}_1$, and P1 outputs $\mathbf{x}_j$ in $\mathcal{H}_5$.

Thus the claim follows from the fact that $\mathcal{E}$ occurs with negligible probability in $\mathcal{H}_5$, which follows from Lemma 9. □

Lemma 11. *Assuming security of the garbled circuit scheme and statistical sender's security of the OT, the view of $\mathcal{A}$ between $\mathcal{H}_5$ and $\mathcal{H}_6$ is computationally indistinguishable.*

Proof. We consider a subhybrid $\mathcal{H}'_5$ which acts similarly to $\mathcal{H}_5$ except that when generating the second-round message for each honest P_i during session (s, i, j), it uses $\mathsf{onlychoices}$ to zero out the labels given by the honest party in $\mathbf{ot}_2$ which do not correspond to the adversary's input.

This claim then then follows from the next two claims. □

Lemma 12. *Assuming statistical sender's security of the OT, the view of the adversary $\mathcal{A}$ between $\mathcal{H}_5$ and $\mathcal{H}'_5$ are statistically indistinguishable.*

Proof. We prove the claim via a sequence of subhybrids for each session (s, i, j), where in each subhybrid we switch the $\mathbf{ot}_2$ message of P_i to zero out non-chosen labels. Assume there is an adversary $\mathcal{A}$ who can interact with the simulator and then distinguish between the subhybrid for some session (s, i, j) and the preceding subhybrid with nonnegligible probability. We use $\mathcal{A}$ to build an adversary $\mathcal{A}'$ for the OT sender's security game.

Fix the randomness $r_{gc}^{P_i}$ used to generate P_i's garbled circuit which it sends as part of its second-round message. There must be at least one such fixed value for which $\mathcal{A}$ still distinguishes with nonnegligible probability. Let $\mathcal{A}'$ run the experiment identically to the subhybrid preceding (s, i, j) with $r_{gc}^{P_i}$ fixed to this value, except that it passes the $\mathbf{ot}_1$ message $\mathbf{ot}_1^{P_j}$ generated by $\mathcal{A}$ to the OT challenger. The OT challenger then either responds with an $\mathbf{ot}_2^{P_i}$ corresponding to the same labels in $\mathcal{H}_5$, or breaks $\mathbf{ot}_1^{P_j}$ and zeros out the labels which do not correspond to the adversary's input. $\mathcal{A}'$ then outputs the output of $\mathcal{A}$.

Assuming $r_{gc}^{P_i}$ is fixed, this experiment is identical to the preceding hybrid in the first case and the subhybrid for (s, i, j) in the second case. So if $\mathcal{A}$ successfully distinguishes then $\mathcal{A}'$ does as well. □

Lemma 13. *Assuming security of the garbled circuit scheme, the views induced by $\mathcal{H}'_5$ and $\mathcal{H}_6$ are computationally indistinguishable.*

Proof. We prove the claim via a sequence of subhybrids for each session (s, i, j), where in each subhybrid we switch the garbled circuit of P_i to be simulated.

Assume there is an adversary $\mathcal{A}$ who distinguishes between the subhybrid directly preceding the one for some session (s, i, j) and the subhybrid for (s, i, j)

with nonnegligible probability. We use $\mathcal{A}$ to build a $\mathsf{poly}(T_{\mathsf{OT}_{\mathsf{extract}}})$-time adversary $\mathcal{A}'$ that contradicts security of the garbled circuit.

Fix the randomness used by $\mathcal{A}$ and the randomness used by the simulator in generating all rounds preceding P_i's second-round message during session (s, i, j). There must be some such fixed randomness such that $\mathcal{A}$ still distinguishes with nonnegligible probability. This also fixes the circuit which the honest party P_i garbles along with P_j's $\mathbf{ot}_1$ input in session (s, i, j).

Let $\mathcal{A}'$ work in the same way as the subhybrid preceding (s, i, j) except that it receives a garbled circuit and labels $(\tilde{C}^{\mathsf{P}_1}, \mathit{labels})$ from the challenger, which it uses as the garbled circuit and labels for P_i in session (s, i, j). The challenger either computes the garbled circuit honestly or simulates using the output $f(\mathbf{x}_i, \mathbf{x}_j)$, which is fixed because of the fixed randomness. $\mathcal{A}'$ finally outputs the output of $\mathcal{A}$. Based on what the challenger does, the experiment is either identical to the subhybrid preceding (s, i, j) or the (s, i, j) subhybrid (with the adversary's randomness fixed). This means that $\mathcal{A}$ distinguishes with nonnegligible probability and thus so does $\mathcal{A}'$, contradicting security of the garbled circuit, since $\mathsf{poly}(T_{\mathsf{OT}_{\mathsf{extract}}}) \ll T_G$. □

4 Instantiating the Non-Malleable Commitment of [51] using LWE

In this section, we list all primitives used in each part of the construction of fully concurrent non-malleable commitments of [51], along with how to instantiate each primitive using subexponential hardness of LWE.

Two-round extractable commitments are constructed using the following primitives:
- A non-interactive perfectly binding commitment. *Can be obtained from LWE using one of the LWE-based PKE schemes.*
- Two-round statistically-sender-private oblivious transfer. *Known from LWE [19].*
- Yao's garbled circuits. *Symmetric-key encryption is known from LWE.*
- Two-round zero knowledge with super-polynomial simulation. *Known from LWE [8, 43].*

Two-round SPS strong zero knowledge arguments are constructed using the following primitives:
- A zap. *Known from LWE [8, 43].*
- The two-round extractable commitment. *See above.*
- A trapdoor for the prover to use. *Can use an instance of SIS.*

Two-round constant-tag non-malleable commitments are constructed using the following primitives:
- The two-round extractable commitment. *See above.*
- A non-interactive perfectly binding commitment. *Can be obtained from LWE using one of the LWE-based PKE schemes.*

One-one non-malleable commitments in two rounds are constructed using the following primitives:

- The two-round constant-tag non-malleable commitment. *See above.*
- The two-round SPS strong ZK. *See above.*

One-one simulation-sound zero knowledge in two rounds is constructed using the following primitives:

- The one-one non-malleable commitment. *See above.*
- A zap. *Known from LWE [8, 43].*
- A trapdoor for the prover to use. *Can use an instance of SIS.*

Fully-concurrent non-malleable commitments in two rounds are constructed using the following primitives:

- The one-one simulation-sound zero knowledge argument. *See above.*
- The constant-tag non-malleable commitment. *See above.*

5 Quantum Computation

We conclude by observing that the simulator for our two-party computation protocol is straight-line, i.e., it does not resort to rewinding the adversary to generate a simulated transcript, and black-box. It is shown in [69] that any such protocol remains secure also against quantum attackers (i.e., it is post-quantum secure) and furthermore in [14] it is shown that any post-quantum two-round 2PC with a straight-line black-box simulator can be generically compiled into a secure 2PC for quantum circuits without adding any round. In [14] they proposed an instantiation of the classical 2PC in the common reference string model. Plugging our protocol (with a suitable instantiation of the underlying building blocks) into their result we obtain the following new implication.

Theorem 3. *Assuming the quantum sub-exponential hardness of the LWE problem, there exists a two-round concurrent 2PC for all quantum circuits, with classical inputs and outputs, in the plain model.*

5.1 Quantum Concurrent 2PC

For completeness, we describe our construction of concurrently secure two-round 2PC with respect to quantum functionalities. Our protocol is essentially identical to the three-message two-party protocol described in [14], except that we substitute our (classical) 2PC protocol from Section 3 and we only require one party to know the output. Hence we drop the third round from the protocol described in [14]. Before describing the protocol, we need bring some definitions of the quantum primitives and we take them in verbatim from [14].

Definition 5 (Clifford + Measurement Circuit). *A Clifford + Measurement circuit with parameters $n_i + k_{i i\in[d]}$ operates on $n_1 + k_1$ input qubits and applies d alternating layers of Clifford unitary and computational basis measuremtns, during which a total of $k_1 + \cdots + k_d$ of the input qubits are measured. It is specified by $(F_0, f_1, \ldots, f_d)$, where F_0 us a Clifford unitary, and each f_i is a classical circuit which takes as input the result of computational basis measurements on the i-th layer, and outputs a Clifford unitary F_i. In layer $i \in [d]$, k_i*

qubits are measured and n_i qubits are left over. The circuit is evaluated by first applying F_0 to the $n_1 + k_1$ input qubits, then the following steps are performed for $i = 1, \ldots d$:

- *Measure the remaining k_i qubits in the computational basis, resulting in outcomes $m_i \in \{0,1\}^{k_1}$.*
- *Evaluate $f_i(m_i)$ to obtain a classical description of a Clifford $F_i \in \mathfrak{C}_{n_i}$.*
- *Apply F_i to the first n_i registers.*

The output of the circuit is the result of applying F_d to the final n_d registers.

Definition 6 (Garbling scheme for $\mathsf{C}+\mathsf{M}$ Circuits). *A garbling scheme for $\mathsf{C}+\mathsf{M}$ circuits consists of three procedures $(\mathtt{QGarble}, \mathtt{QGEval}, \mathtt{QGSim})$ with the following syntax.*

- *$(E_0, \tilde{Q}) \leftarrow \mathtt{QGarble}(1^\lambda, Q)$: A classical PPT procedure that takes as input the security parameter and a $\mathsf{C}+\mathsf{M}$ circuit and outputs a Clifford input garbling matrix E_0 and a quantum garbled circuit $\tilde{Q}$.*
- *$\mathbf{x}_{out} \leftarrow \mathtt{QGEval}(\tilde{\mathbf{x}}_{inp}, \tilde{Q})$: A QPT procedure that takes as input a garbled input $\tilde{\mathbf{x}}_{inp}$ and a garbled $\mathsf{C}+\mathsf{M}$ circuit $\tilde{Q}$, and outputs a quantum state $\mathbf{x}_{out}$.*
- *$(\tilde{x}_{inp}, \tilde{Q}) \leftarrow \mathtt{QGSim}(1^\lambda, \{n_i, k_i\}_{i\in[d]}, \mathbf{x}_{out})$: A QPT procedure that takes as input the security parameter, parameters for a $\mathsf{C}+\mathsf{M}$ circuit, and an output state, and outputs a simulated garbled input and garbled circuit.*

Correctness. For any $\mathsf{C}+\mathsf{M}$ circuit Q with parameters $\{n_i + k_i\}_{i\in[d]}$ and n_0-qubit input state $\mathbf{x}_{\text{inp}}$ along with (potentially entangled) auxiliary information z, we have:

$$\{(\mathtt{QGEval}(E_0(\mathbf{x}_{\text{inp}}, 0^{k\lambda}), \tilde{Q}), z) : (E_0, \tilde{Q}) \leftarrow \mathtt{QGarble}(1^\lambda, Q)\} \approx_s (Q(\mathbf{x}_{\text{inp}}), z)).$$

Security. For any $\mathsf{C}+\mathsf{M}$ circuit Q with parameters $\{n_i + k_i\}_{i\in[d]}$ and n_0-qubit input state x_{inp} along with (potentially entangled) auxiliary information z, we have:

$$\begin{aligned}\{(E_0(\mathbf{x}_{\text{inp}}, 0^{k\lambda}), \tilde{Q}, z) : (E_0, \tilde{Q}) \leftarrow \mathtt{QGarble}(1^\lambda, Q)\}\\ \approx_c (\mathtt{QGSim}(1^\lambda, \{n_i, k_i\}_{i\in[d]}, Q(\mathbf{x}_{\text{inp}})), z).\end{aligned}$$

The rest of this section assumes familiarity with basic notions of quantum computation, and quantum garbled circuits as we defined above and were initially introduced in [24]. For a comprehensive background, we refer the reader to [14]. For the construction of our protocols, we need the following building blocks:

i A post-quantum secure concurrent two-round two-party protocol for classical computation $\mathtt{c2PC}$ (let $T_{\mathtt{c2PC}}$ denote the runtime of the simulator).
ii A quantum garbling scheme for $\mathsf{C}+\mathsf{M}$ gates $(\mathtt{QGarble}, \mathtt{QGEval}, \mathtt{QGSim})$, secure against adversaries running in time $\mathsf{T}_{\mathsf{QGC}}$.

We denote by T_{c2PC} the runtime of the simulator of the classical two party computation and by T_{QGC} the maximum runtime of the distinguisher allowed by the security of the quantum garbled circuit. We require that $\mathsf{poly}(\lambda) \ll T_{\mathsf{c2PC}} \ll T_{\mathsf{QGC}}$. In our protocol, parties P_1 and P_2 each having inputs $\mathbf{x}_0$ and $\mathbf{x}_1$ want to compute a quantum circuit Q on their inputs and only P_1 gets outputs out_1 while P_2 receives no output in a way that they do not reveal anything about their inputs and only one party gets the output of the computation. We assume that Q is a Clifford+Measurment quantum circuit and takes input $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{T}^k, \mathbf{0}^k)$, where $\mathbf{T}$ denotes a *magic* state $\mathbf{T} = 1/\sqrt{2} \cdot (|0\rangle + e^{i\pi/4} |1\rangle)$.

At a high level, the quantum two-party protocol between parties $\mathsf{P}_1, \mathsf{P}_2$ runs in two parallel phases, one is for the parties to jointly encode their quantum inputs and simultaneously they run a two-message classical 2PC that outputs a classical description of a quantum garble circuit to P_1 according to the functionality $f[Q]$ which is described in fig. 2; which allows P_1 to later evaluate the garbled circuit to get his own output.

To make the protocol secure against a malicious P_2 we use the "cut and choose" technique from [32] and we refer to [14, §2,5] where they explained how this technique would be applied on their two-party protocol to make it secure against malicious P_2. The cut-and-choose technique is done by the Clifford unitary $\mathrm{U}_{\text{dec-check-enc}}$ in the functionality of our $\mathsf{c2PC}$ that is used in the computation phase of our protocol below. Now we describe our protocol in more detail:

Quantum two party computation protocol

Common information: security parameter λ, and $\mathsf{C}+\mathsf{M}$ circuit Q to be computed with $n_1 + n_2$ input qubits, $m_1 + m_2$ output qubits, n_Z auxiliary $\mathbf{0}$ states, and n_T auxiliary T states. let
$s = n_2 + (n_1 + \lambda) + (2n_Z + \lambda) + (n_T + 1)\lambda$.
P_1**'s Input:** $\mathbf{x}_1$ (Parsed as a computational basis state)
P_2**'s Input:** $\mathbf{x}_2$ (Parsed as a computational basis state)

- **Round 1** P_1 :
  - Samples a random Clifford $C_1 \leftarrow \mathfrak{C}_{n_1+\lambda}$ and uses it to encrypt and authenticate his input $\mathbf{x}_1$ as $\mathbf{m}_1 \coloneqq C_1(\mathbf{x}_1, \mathbf{0}^\lambda)$.
  - Computes the first round message m_1, of the $\mathsf{c2PC}$ using C_1 as his input.
  - Sends $(m_1, \mathbf{m}_1)$ to P_2.
- **Round 2** P_2 :
  - Samples random Cliffords $C_2 \leftarrow \mathfrak{C}_s$ and $C_{out} \leftarrow \mathfrak{C}_{m_2+\lambda}$ uses C_2 to encrypt and authenticate his own input $\mathbf{x}_2$ alongside with encoding of his quantum state $\mathbf{m}_2 \coloneqq C_2(\mathbf{x}_2, \mathbf{m}_1, \mathbf{0}^{2n_Z}, T^{(n_T+1)\lambda})$.
  - Computes the second round message m_2 for the classical $\mathsf{c2PC}$ computation using (C_2, C_{out}) as their input.

- Sends $(m_2, \mathbf{m}_2)$ to P_1.

– **Computation Phase** P_1 does the following computation:
- Using m_2 he can compute the output of the classical `c2PC` that is $(\mathrm{U}_{\text{dec-check-enc}}, D, \tilde{g}_1, \ldots, \tilde{g}_d)$.
- Compute $(\mathbf{m}_{\text{inp}}, \mathbf{z}_{\text{check}}, \text{trap}_2, \mathrm{T}_{\text{inp}}, \mathbf{t}_{\text{check}}) \leftarrow \mathrm{U}_{\text{dec-check-enc}}(\mathbf{m}_2)$
- Measure each qubit of $(\mathbf{z}_{\text{check}}, \text{trap})$ in the standard basis and abort if any measurement is not zero.
- Measure each qubit of $\mathbf{t}_{\text{check}}$ and the T-basis and abort if any measurement is not zero.
- Compute $\text{out}_1 \leftarrow \texttt{QGEval}(D_0, \tilde{g}_1, \ldots, \tilde{g}_d, \mathbf{m}_{\text{inp}})$.

Classical functionality $f[Q]$

Common Information: security parameter λ, and $\mathsf{C}+\mathsf{M}$ circuit Q to be computed with $n_1 + n_2$ input qubits, $m_1 + m_2$ output qubits, n_Z auxiliary $\mathbf{0}$ states, and n_T auxiliary T states. let $s = n_2 + (n_1 + \lambda) + (2n_Z + \lambda) + (n_T + 1)\lambda$.

P_1**'s Input:** Classical description of $C_1 \in \mathfrak{C}_{n_1+\lambda}$
P_2**'s Input:** Classical description of $C_2 \in \mathfrak{C}_s$ and $C_{\text{out}} \in \mathfrak{C}_{m_2+\lambda}$

1. Sample the unitary U_{check} as follows:
   – sample random permutation π on $(n_T + 1)\lambda$ elements.
   – Sample random element $M \leftarrow \mathrm{GL}(2n_T, \mathbb{F}_2)$.
   – Compute a description of the Clifford U_{check} that operates as follows on registers $(\mathrm{A}, \mathrm{B}, \mathrm{Trap}_{\mathrm{B}}, \mathrm{Z}_{\mathrm{A}}, \mathrm{Trap}_{\mathrm{A}}, \mathrm{T}_{\mathrm{A}})$.
     - Rearrange the registers of T_{A} according to the permutation π and then partition the registers into $(\mathrm{T}_{\text{inp}}, \mathrm{T}_{\text{check}})$.
     - Apply the inner map M to the registers Z_{A} and then partition the registers into $(\mathrm{Z}_{\text{inp}}, \mathrm{Z}_{\text{check}})$.
     - Rearrange the registers to $(\mathrm{A}, \mathrm{B}, \mathrm{Z}_{\text{inp}}, \mathrm{Trap}_{\mathrm{A}}, \mathrm{T}_{\text{inp}}, \mathrm{Z}_{\text{check}}, \mathrm{Trap}_{\mathrm{B}}, \mathrm{T}_{\text{check}})$.
   – Define $U_{\text{dec-check}}$ as:

$$U_{\text{dec-check}} \coloneqq U_{\text{check}}(\mathbb{I}^{n_A} \otimes C_B^\dagger \otimes \mathbb{I}^{(2n_Z+\lambda)+(n_T+1)\lambda})C_2^\dagger$$

2. Sample $(E_0, D_0, \tilde{g}_1, \cdots, \tilde{g}_d) \leftarrow \texttt{QGarble}(1^\lambda, Q_{\text{dist}}(C_{\text{out}}))$.
3. Compute the description of $\mathrm{U}_{\text{dec-check-enc}} \coloneqq (E_0 \otimes \mathbb{I}^{(n_Z+\lambda)\lambda})U_{\text{dec-check}}^\dagger$.

P_1 **Output:**

1. A unitary $\mathrm{U}_{\text{dec-check-enc}}$ on s qubits.
2. A quantum garbled circuit $(D, \tilde{g}_1, \ldots, \tilde{g}_n)$

Fig. 2. Classical functionality $\mathcal{F}[Q]$ to be used in the quantum `c2PC` protocol

We note that, in step 2 of fig. 2, given a $\mathsf{C}+\mathsf{M}$ circuit Q, and a Clifford $C_{\text{out}} \in \mathfrak{C}_{m_1+\lambda}$, the circuit $Q_{\text{dist}}(C_{\text{out}})$ is defined as follows: it takes as input $(n_1+n_2+n_Z+\lambda+n_T\lambda)$ qubits $(\mathbf{x}_1+\mathbf{x}_2+z_{\text{inp}}+trap_A, t_{\text{inp}})$ on registers $(A, B, Z_{\text{inp}}, Trap_A, T_{\text{inp}})$, it will first apply the magic state distillation circuit from Lemma 3.3 of [14] with parameters $(n_T\lambda, \lambda)$ to t_{inp} to produce QRV t of size n_T, then it will run Q on $(\mathbf{x}_1, \mathbf{x}_2, z_{\text{inp}}, t)$ to produce (y_1, y_2), and in the end, it outputs $(C_{\text{out}}(y_2, trap_A), y_1)$.

5.2 Security Proof of Quantum Concurrent 2PC

In this section, we prove Lemma 14 that is similar to the proof of Theorem 5.1 in [14]. For the sake of completeness, we write out the proof in its entirety.

Lemma 14. *Assuming two-round concurrently secure 2PC protocol with black-box super-polynomial simulation running in time T_{c2PC} and sub-exponentially secure quantum garbled circuit with simulation running time T_{QGC}, where $\mathsf{poly}(\lambda) \ll T_{\mathsf{c2PC}} \ll T_{\mathsf{QGC}}$, there exists a concurrent two-round 2PC for all quantum circuits in the plain model.*

Proof. For the sake of simplicity, we prove it in the following two cases when

1. A quantum polynomial-time adversary $\mathcal{A}$ corrupting party P_1.
2. A quantum polynomial-time adversary $\mathcal{A}$ corrupting party P_2.

For the ideal functionality, we use the Classical functionality in fig. 2.

Case 1: Consider any quantum PT adversary $\mathcal{A}$ corrupting party P_1. The simulator $\mathsf{Sim}(\mathbf{x}_1, \mathsf{aux}_{\mathcal{A}})$ is defined as follows:

- Receive $(m_1, \mathbf{m}_1)$ from $\mathcal{A}$ and compute $\mathsf{inp} \leftarrow \mathsf{c2PC.Sim}(1^\lambda, m_1)$. If $\mathsf{inp} = \perp$ the abort, else parse inp as C_1 and compute $(\mathbf{x}_1', \text{trap}_1) := C_1^\dagger(\mathbf{m}_1)$.

- Query ideal functionality and compute simulated round 2 message as follows:
  - Compute $(\tilde{\mathbf{m}}_{\text{inp}}, D, \tilde{g}_1, \ldots, \tilde{g_d})$ by running $\mathtt{QGSim}(1^\lambda, \{n_i, k_i\}_{i\in[d]}, \text{out}_1)$ where where $\tilde{\mathbf{m}}_{\text{inp}}$ is the simulated quantum garbled input and $\{n_i, k_i\}_{i\in[d]}$ are the parameters of $\mathsf{C}+\mathsf{M}$ circuit $Q_{\text{dist}}(C_{\text{out}})$.
  - Sample a random $\text{U}_{\text{dec-check-enc}}$ and compute $\mathbf{m}_2 \coloneqq \text{U}^\dagger_{\text{dec-check-enc}}(\tilde{\mathbf{m}}_{\text{inp}}, \mathbf{0}, \text{trap}_1, T^\lambda)$.
  - Compute $m_2 \leftarrow \mathsf{c2PC.Sim}(1^\lambda, \text{U}_{\text{dec-check-enc}}, D, \tilde{g}_1, \ldots, \tilde{g}_d)$.
  - Send $(m_2, \mathbf{m}_2)$ to $\mathcal{A}$.

We now show that the simulation strategy is successful against all malicious QPT adversaries. That is, the view of the adversary $\mathcal{A}(\mathbf{x}_2, \mathsf{aux}_{\mathcal{A}})$ along with the output of the honest parties is computationally indistinguishable in the real and ideal worlds. We show this via a series of computationally indistinguishable hybrids where the first world $\mathcal{H}_0$ corresponds to the real world and the last game corresponds to the ideal world.

1. $\mathcal{H}_1$: In this game, Sim runs $\mathtt{c2PC}.\mathsf{Sim}_{\mathsf{P}_1}$ in time $T_{\mathtt{c2PC}}$ and simulates the $\mathtt{c2PC}$ scheme, using $\mathtt{c2PC}.\mathsf{Sim}$ to extract $\mathcal{A}$'s input, C_1, and runs $\mathtt{c2PC}.\mathsf{Sim}$ to computes party P_2's message m_2. Use C_1 and freshly sampled (C_2, C_{out}) to sample the output of the classical functionality that is given to $\mathtt{c2PC}.\mathsf{Sim}$. The (computational) indistinguishability of $\mathcal{H}_0$ and $\mathcal{H}_1$ comes directly from the security against corrupted $\mathtt{c2PC}$ scheme.

2. $\mathcal{H}_2$: Now, we make a (perfectly indistinguishable) switch in how $\mathbf{m}_2$ is computed and how $\mathrm{U}_{\text{dec-check-enc}}$ (part of the classical $\mathtt{c2PC}$ output) is sampled. Define $(\mathbf{x}'_1, \text{trap}_1) := C_1^\dagger(\mathbf{m}_1)$, where C_1 was extracted from m_1. As here exists a Clifford unitary U such that $\mathrm{U}_{\text{dec-check-enc}} = UC_2^\dagger$, where C_2 was randomly sampled. Thus, since the Clifford matrices form a group, an equivalent sampling procedure would be to sample $\mathrm{U}_{\text{dec-check-enc}}$ and define

$$\mathbf{m}_2 := \mathrm{U}^\dagger_{\text{dec-check-enc}}(E_0(\mathbf{x}'_1, \mathbf{x}_2, \mathbf{0}^{n_Z+\lambda}, T^{n_T\lambda}), \mathbf{0}^{n_Z}, \text{trap}_1, T^\lambda).$$

   Notice that, in $\mathcal{H}_1$, we have that,

$$\mathrm{U}_{\text{dec-check-enc}}(\mathbf{m}_2) \coloneqq (E_0(\mathbf{x}'_1, \mathbf{x}_2, \mathbf{0}^{n_Z+\lambda}, T^{n_T\lambda}), \mathbf{0}^{n_Z}, \text{trap}_1, T^\lambda).$$

   This hybrid runs in time $T_{\mathtt{c2PC}}$. We observe that $\mathcal{H}_1$ and $\mathcal{H}_2$ are equivalent.

3. $\mathcal{H}_3$: In this game, we simulate the quantum garbled circuit. In particular, compute

$$\text{out}_1 \leftarrow \mathsf{Q}_{\mathsf{dist}}[C_{\text{out}}](\mathbf{x}'_1, \mathbf{x}_2, \mathbf{0}^{n_Z+\lambda}, T^{n_T\lambda})$$

   and then compute $\hat{\mathbf{m}}_{\text{inp}}$ by running $\mathtt{QGSim}$ and the substitute $\hat{\mathbf{m}}_{\text{inp}}$ for $E_0(\mathbf{x}'_1, \mathbf{x}_2, \mathbf{0}^{n_Z+\lambda}, T^{n_T\lambda})$ in the computation of $\mathbf{m}_2$, so that $\mathbf{m}_2 := \mathrm{U}^\dagger_{\text{dec-check-enc}}$ $(E_0(\mathbf{x}'_1, \mathbf{x}_2, \mathbf{0}^{n_Z+\lambda}, T^{n_T\lambda}), \mathbf{0}^{n_Z}, \text{trap}_1, T^\lambda)$. This hybrid runs in time $T_{\mathtt{c2PC}}$. The (computational) indistinguishability of $\mathcal{H}_2$ and $\mathcal{H}_3$ comes directly from the sub-exponential security of the QGC.

4. $\mathcal{H}_4$: Finally, instead of directly computing out_1 from the first stage of $\mathsf{Q}_{\mathsf{dist}}$, query the ideal functionality with $\mathbf{x}'_1$ and receive back out_1. Now, during party P_2's output reconstruction step, if the check passes, send "accept" to the ideal functionality, and otherwise send "abort" to the ideal functionality. This game is now same as the ideal world. This hybrid runs in $T_{\mathtt{c2PC}}$.
   We observe that $\mathcal{H}_4$ and $\mathcal{H}_5$ are equivalent .

Case 2: Consider any quantum PT adversary $\mathcal{A}$ corrupting party P_2. The simulator Sim is defined as follows. Whenever we say that the simulator aborts, we mean that it sends $\perp$ to the ideal functionality and the adversary. The simulator $\mathsf{Sim}(\mathbf{x}_2, \mathsf{aux}_{\mathcal{A}})$ works as follows:

– Compute $m_1 \leftarrow \mathtt{c2PC}.\mathsf{Sim}(1^\lambda)$, samples a random Clifford C_1, and compute $\mathbf{m}_1 := C_1(\mathbf{0}^{n_1}, \mathbf{0}^\lambda)$. Send $(m_1, \mathbf{m}_1)$ to $\mathcal{A}(\mathbf{x}_2, \mathsf{aux}_{\mathcal{A}})$.

– Receive $(m_2, \mathbf{m}_2)$ from $\mathcal{A}$ and compute $\mathsf{out} \leftarrow \mathsf{c2PC.Sim}(1^\lambda, m_2)$. Abort if $\mathsf{out} = \bot$, otherwise, parse out as (C_2, C_{out}).

– Using (C_1, C_2) sample $U_{\text{dec-check}}$ and compute $(\mathbf{x}'_1, \mathbf{x}'_2, \mathbf{m}_{\text{inp}}, \text{trap}_2, \mathbf{z}_{\text{check}}, \text{trap}_1, \text{T}_{\text{inp}}, \mathbf{t}_{\text{check}}) \leftarrow U_{\text{dec-check}}(\mathbf{m}_2)$. Measure each qubit of $\mathbf{z}_{\text{check}}$ and trap_2 in the standard basis and each qubit of $\mathbf{t}_{\text{check}}$ in the T-basis. If any measurement is non-zero, then abort.

We observe that the simulation strategy is successful against all malicious QPT adversaries. That is, the view of the adversary along with the output of the honest parties is computationally indistinguishable in the real and ideal worlds. For this, we consider the following hybrids where $\mathcal{H}_0$ is the real word.

1. $\mathcal{H}_1$ In this world we run $\mathsf{c2PC.Sim}$ to compute the first message of the $\mathsf{c2PC}$ as m_1 and extract (C_2, C_{out}) (or abort). This hybrid runs in time T_{c2PC}. This world is computationally indistinguishable from the real world by the security of the $\mathsf{c2PC}$ scheme against the malicious P_2.
2. $\mathcal{H}_2$ In this world, we compute $\mathbf{m}_1$ as $C_1(\mathbf{0}^{n_B}, \mathbf{0}^\lambda)$ and substitute $\mathbf{x}_1$ with $\mathbf{x}'_1$ before computing $Q_{\text{dist}}(C_{\text{out}})$. This hybrid runs in time T_{c2PC} This world is statistically indistinguishable from $\mathcal{H}_1$ from the security of the Clifford authentication code.

□

Acknowledgements. Giulio Malavolta was supported by the German Federal Ministry of Education and Research BMBF (grant 16KISK038, project 6GEM). Behzad Abdolmaleki was supported by the German Federal Ministry of Education and Research BMBF (grant 16KISK038, project 6GEM) and most of the work was done while he was affiliated with the Max Planck Institute for Security and Privacy. Rex Fernando was supported by the Algorand Centres of Excellence (ACE) Programme, the Defense Advanced Research Projects Agency under award number HR001120C0086, the Office of Naval Research under award number N000142212064, and the National Science Foundation under award numbers 2128519 and 2044679. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

References

1. Amit Agarwal, James Bartusek, Vipul Goyal, Dakshita Khurana, and Giulio Malavolta. Two-round maliciously secure computation with super-polynomial simulation. In *TCC*, 2021.
2. Shweta Agrawal. Indistinguishability obfuscation without multilinear maps: New methods for bootstrapping and instantiation. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 191–225. Springer, Heidelberg, May 2019.

3. Shweta Agrawal and Alice Pellet-Mary. Indistinguishability obfuscation without maps: Attacks and fixes for noisy linear FE. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 110–140. Springer, Heidelberg, May 2020.
4. Prabhanjan Ananth, Arka Rai Choudhuri, and Abhishek Jain. A new approach to round-optimal secure multiparty computation. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 468–499. Springer, Heidelberg, August 2017.
5. Prabhanjan Ananth, Aayush Jain, and Amit Sahai. Indistinguishability obfuscation without multilinear maps: iO from LWE, bilinear maps, and weak pseudorandomness. Cryptology ePrint Archive, Report 2018/615, 2018. `https://eprint.iacr.org/2018/615`.
6. Prabhanjan Ananth and Amit Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 152–181. Springer, Heidelberg, April / May 2017.
7. Prabhanjan Vijendra Ananth, Divya Gupta, Yuval Ishai, and Amit Sahai. Optimizing obfuscation: Avoiding Barrington's theorem. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014*, pages 646–658. ACM Press, November 2014.
8. Saikrishna Badrinarayanan, Rex Fernando, Aayush Jain, Dakshita Khurana, and Amit Sahai. Statistical ZAP arguments. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 642–667. Springer, Heidelberg, May 2020.
9. Saikrishna Badrinarayanan, Sanjam Garg, Yuval Ishai, Amit Sahai, and Akshay Wadia. Two-message witness indistinguishability and secure computation in the plain model from new assumptions. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part III*, volume 10626 of *LNCS*, pages 275–303. Springer, Heidelberg, December 2017.
10. Saikrishna Badrinarayanan, Vipul Goyal, Abhishek Jain, Yael Tauman Kalai, Dakshita Khurana, and Amit Sahai. Promise zero knowledge and its applications to round optimal MPC. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 459–487. Springer, Heidelberg, August 2018.
11. Saikrishna Badrinarayanan, Vipul Goyal, Abhishek Jain, Dakshita Khurana, and Amit Sahai. Round optimal concurrent MPC via strong simulation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 743–775. Springer, Heidelberg, November 2017.
12. Saikrishna Badrinarayanan, Eric Miles, Amit Sahai, and Mark Zhandry. Post-zeroizing obfuscation: New mathematical tools, and the case of evasive circuits. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 764–791. Springer, Heidelberg, May 2016.
13. Boaz Barak, Manoj Prabhakaran, and Amit Sahai. Concurrent non-malleable zero knowledge. In *47th FOCS*, pages 345–354. IEEE Computer Society Press, October 2006.
14. James Bartusek, Andrea Coladangelo, Dakshita Khurana, and Fermi Ma. On the round complexity of secure quantum computation. Cryptology ePrint Archive, Report 2020/1471, 2020. `https://ia.cr/2020/1471`.
15. James Bartusek, Yuval Ishai, Aayush Jain, Fermi Ma, Amit Sahai, and Mark Zhandry. Affine determinant programs: A framework for obfuscation and witness

encryption. In Thomas Vidick, editor, *ITCS 2020*, volume 151, pages 82:1–82:39. LIPIcs, January 2020.
16. Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 139–155. Springer, Heidelberg, May 2000.
17. Steven M. Bellovin and Michael Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *1992 IEEE Symposium on Security and Privacy*, pages 72–84. IEEE Computer Society Press, May 1992.
18. Nir Bitansky and Huijia Lin. One-message zero knowledge and non-malleable commitments. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part I*, volume 11239 of *LNCS*, pages 209–234. Springer, Heidelberg, November 2018.
19. Zvika Brakerski and Nico Döttling. Two-message statistically sender-private OT from LWE. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 370–390. Springer, Heidelberg, November 2018.
20. Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Candidate iO from homomorphic encryption schemes. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 79–109. Springer, Heidelberg, May 2020.
21. Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Factoring and pairings are not necessary for iO: Circular-secure LWE suffices. Cryptology ePrint Archive, Report 2020/1024, 2020. `https://eprint.iacr.org/2020/1024`.
22. Zvika Brakerski, Shai Halevi, and Antigoni Polychroniadou. Four round secure computation without setup. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 645–677. Springer, Heidelberg, November 2017.
23. Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 1–25. Springer, Heidelberg, February 2014.
24. Zvika Brakerski and Henry Yuen. Quantum garbled circuits, 2020.
25. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.
26. Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip D. MacKenzie. Universally composable password-based key exchange. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 404–421. Springer, Heidelberg, May 2005.
27. Ran Canetti, Huijia Lin, and Rafael Pass. Adaptive hardness and composable security in the plain model from standard assumptions. In *51st FOCS*, pages 541–550. IEEE Computer Society Press, October 2010.
28. Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 3–12. Springer, Heidelberg, April 2015.
29. Arka Rai Choudhuri, Michele Ciampi, Vipul Goyal, Abhishek Jain, and Rafail Ostrovsky. Round optimal secure multiparty computation from minimal assumptions. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 291–319. Springer, Heidelberg, November 2020.
30. Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrède Lepoint, Hemanta K. Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. Zeroizing

without low-level zeroes: New MMAP attacks and their limitations. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 247–266. Springer, Heidelberg, August 2015.
31. Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *23rd ACM STOC*, pages 542–552. ACM Press, May 1991.
32. Yfke Dulek, Alex B. Grilo, Stacey Jeffery, Christian Majenz, and Christian Schaffner. Secure multi-party quantum computation with a dishonest majority. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 729–758. Springer, Heidelberg, May 2020.
33. Rex Fernando, Aayush Jain, and Ilan Komargodski. Maliciously-secure mrnisc in the plain model. Cryptology ePrint Archive, Report 2021/1319, 2021. `https://ia.cr/2021/1319`.
34. Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.
35. Sanjam Garg, Vipul Goyal, Abhishek Jain, and Amit Sahai. Concurrently secure computation in constant rounds. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 99–116. Springer, Heidelberg, April 2012.
36. Sanjam Garg, Susumu Kiyoshima, and Omkant Pandey. On the exact round complexity of self-composable two-party computation. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 194–224. Springer, Heidelberg, April / May 2017.
37. Sanjam Garg, Pratyay Mukherjee, Omkant Pandey, and Antigoni Polychroniadou. The exact round complexity of secure computation. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 448–476. Springer, Heidelberg, May 2016.
38. Sanjam Garg and Akshayaram Srinivasan. Garbled protocols and two-round MPC from bilinear maps. In Chris Umans, editor, *58th FOCS*, pages 588–599. IEEE Computer Society Press, October 2017.
39. Romain Gay and Rafael Pass. Indistinguishability obfuscation from circular security. Cryptology ePrint Archive, Report 2020/1010, 2020. `https://eprint.iacr.org/2020/1010`.
40. Rosario Gennaro and Yehuda Lindell. A framework for password-based authenticated key exchange. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 524–543. Springer, Heidelberg, May 2003. `https://eprint.iacr.org/2003/032.ps.gz`.
41. Vipul Goyal. Constant round non-malleable protocols using one way functions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 695–704. ACM Press, June 2011.
42. Vipul Goyal and Abhishek Jain. On concurrently secure computation in the multiple ideal query model. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 684–701. Springer, Heidelberg, May 2013.
43. Vipul Goyal, Abhishek Jain, Zhengzhong Jin, and Giulio Malavolta. Statistical zaps and new oblivious transfer protocols. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 668–699. Springer, Heidelberg, May 2020.

44. Vipul Goyal, Huijia Lin, Omkant Pandey, Rafael Pass, and Amit Sahai. Round-efficient concurrently composable secure computation via a robust extraction lemma. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part I*, volume 9014 of *LNCS*, pages 260–289. Springer, Heidelberg, March 2015.
45. Shai Halevi, Carmit Hazay, Antigoni Polychroniadou, and Muthuramakrishnan Venkitasubramaniam. Round-optimal secure multi-party computation. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 488–520. Springer, Heidelberg, August 2018.
46. Sam Hopkins, Aayush Jain, and Huijia Lin. Counterexamples to circular security-based io. In *CRYPTO*, 2021.
47. Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. How to leverage hardness of constant-degree expanding polynomials overa $\mathbb{R}$ to build $i\mathcal{O}$. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 251–281. Springer, Heidelberg, May 2019.
48. Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from LPN over $\mathbb{F}_p$, DLIN, and PRGs in NC^0. *IACR Cryptol. ePrint Arch.*, 2021.
49. Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In *STOC*, pages 60–73. ACM, 2021.
50. Dakshita Khurana. Non-interactive distributional indistinguishability (nidi) and non-malleable commitments, 2021.
51. Dakshita Khurana and Amit Sahai. How to achieve non-malleability in one or two rounds. In Chris Umans, editor, *58th FOCS*, pages 564–575. IEEE Computer Society Press, October 2017.
52. Susumu Kiyoshima. Round-efficient black-box construction of composable multi-party computation. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 351–368. Springer, Heidelberg, August 2014.
53. Susumu Kiyoshima, Yoshifumi Manabe, and Tatsuaki Okamoto. Constant-round black-box construction of composable multi-party computation protocol. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 343–367. Springer, Heidelberg, February 2014.
54. Huijia Lin and Rafael Pass. Black-box constructions of composable protocols without set-up. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 461–478. Springer, Heidelberg, August 2012.
55. Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. Concurrent non-malleable commitments from any one-way function. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 571–588. Springer, Heidelberg, March 2008.
56. Huijia Lin and Stefano Tessaro. Indistinguishability obfuscation from trilinear maps and block-wise local PRGs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 630–660. Springer, Heidelberg, August 2017.
57. Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. In Irit Dinur, editor, *57th FOCS*, pages 11–20. IEEE Computer Society Press, October 2016.
58. Yehuda Lindell. Lower bounds for concurrent self composition. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 203–222. Springer, Heidelberg, February 2004.
59. Tal Malkin, Ryan Moriarty, and Nikolai Yakovenko. Generalized environmental security from number theoretic assumptions. In Shai Halevi and Tal Rabin, editors,

TCC 2006, volume 3876 of *LNCS*, pages 343–359. Springer, Heidelberg, March 2006.
60. Silvio Micali, Rafael Pass, and Alon Rosen. Input-indistinguishable computation. In *47th FOCS*, pages 367–378. IEEE Computer Society Press, October 2006.
61. Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 629–658. Springer, Heidelberg, August 2016.
62. Omkant Pandey, Rafael Pass, and Vinod Vaikuntanathan. Adaptive one-way functions and applications. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 57–74. Springer, Heidelberg, August 2008.
63. Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 160–176. Springer, Heidelberg, May 2003.
64. Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In László Babai, editor, *36th ACM STOC*, pages 232–241. ACM Press, June 2004.
65. Rafael Pass, Huijia Lin, and Muthuramakrishnan Venkitasubramaniam. A unified framework for UC from only OT. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 699–717. Springer, Heidelberg, December 2012.
66. Rafael Pass and Alon Rosen. Concurrent non-malleable commitments. In *46th FOCS*, pages 563–572. IEEE Computer Society Press, October 2005.
67. Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 500–517. Springer, Heidelberg, August 2014.
68. Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, October 1999.
69. Fang Song. A note on quantum security for post-quantum cryptography. In Michele Mosca, editor, *Post-Quantum Cryptography - 6th International Workshop, PQCrypto 2014, Waterloo, ON, Canada, October 1-3, 2014. Proceedings*, volume 8772 of *Lecture Notes in Computer Science*, pages 246–265. Springer, 2014.
70. Hoeteck Wee and Daniel Wichs. Candidate obfuscation via oblivious LWE sampling. Cryptology ePrint Archive, Report 2020/1042, 2020. `https://eprint.iacr.org/2020/1042`.

A Preliminaries, Continued

A.1 Garbled Circuits

A garbled circuit scheme consists of a tuple of PPT algorithms $(\mathsf{Garble}, \mathsf{Eval}, \mathsf{Sim}_{\mathsf{Garble}})$, which work as follows.

- $\mathsf{Garble}(1^\lambda, C; r) \to (\tilde{C}, \mathsf{lab})$ takes as input the security parameter 1^λ and a circuit C with λ input bits, and produces a garbled version $\tilde{C}$ along with the list $\mathsf{lab} := \{\mathsf{lab}_{i,b}\}_{i\in[\lambda],b\in\{0,1\}}$ of two labels per input position, corresponding to each bit.

We denote by lab_x the list $\{\mathsf{lab}_{i,x_i}\}$ corresponding to some input string $x \in \{0,1\}^\lambda$.
- $\mathsf{Eval}(\tilde{C}, \mathsf{lab}_x)$ this is a deterministic algorithm that takes a garbled circuit $\tilde{C}$ and a list $\mathsf{lab}_x = \{\mathsf{lab}_{i,x_i}\}$ of labels corresponding to input x, and outputs the evaluation of $\tilde{C}$ on the input corresponding to the labels.

Definition 7. *A tuple* $(\mathsf{Garble}, \mathsf{Eval}, \mathsf{Sim}_{\mathsf{Garble}})$ *is a* (T, ϵ)*-garbled circuit scheme if the following properties hold:*

- ***Correctness.*** *For all* λ*,* C*, and* $x \in \{0,1\}^\lambda$*,*

$$\Pr\left[\mathsf{Eval}(\tilde{C}, \mathsf{lab}_x) = C(x) \middle| (\tilde{C}, \mathsf{lab}) \leftarrow \mathsf{Garble}(1^\lambda, C)\right] = 1.$$

  (Recall that lab_x *is defined to be* $\{\mathsf{lab}_{i,x_i}\}_{i\in[\lambda]}$*.)*
- (T, ϵ)***-Privacy.*** *There exists a PPT algorithm* $\mathsf{Sim}_{\mathsf{Garble}}$ *which takes as input* $(1^\lambda, |C|, y)$*, where* 1^λ *is the security parameter and* $y = C(x)$ *is some output of* C*, such that the following holds. For all ensembles* $\{C_\lambda, x_\lambda\}_\lambda$ *and time-*$T(\lambda)$ *distinguishers* $\mathcal{D}$*,*

$$\left|\Pr\left[\mathcal{D}(1^\lambda, \tilde{C}_\lambda, \mathsf{lab}_{x_\lambda}) = 1\right] - \Pr\left[\mathcal{D}(1^\lambda, \hat{C}_\lambda, \hat{\mathsf{lab}}_{x_\lambda}) = 1\right]\right| < \epsilon(\lambda),$$

  where $(\tilde{C}_\lambda, \mathsf{lab}) \leftarrow \mathsf{Garble}(1^\lambda, C_\lambda)$ *and* $(\hat{C}_\lambda, \hat{\mathsf{lab}}) \leftarrow \mathsf{Sim}_{\mathsf{Garble}}(1^\lambda, |C_\lambda|, C_\lambda(x_\lambda))$*.*

A.2 Two-Round Non-Malleable Commitments

A two-round simultaneous-message non-malleable commitment scheme consists of a tuple of PPT algorithms $(\mathsf{NMC}_1^{\mathsf{send}}, \mathsf{NMC}_1^{\mathsf{recv}}, \mathsf{NMC}_2^{\mathsf{send}})$, which work as follows.

- $\mathsf{NMC}_1^{\mathsf{send}}(1^\lambda, v; r^{\mathsf{send}}) \to c$ takes as input the security parameter 1^λ and the message v and produces the sender's first message committing to v.
- $\mathsf{NMC}_1^{\mathsf{recv}}(1^\lambda; r^{\mathsf{recv}}) \to m$ takes as input the security parameter 1^λ and produces the receiver's (simultaneous) first round message m.
- $\mathsf{NMC}_2^{\mathsf{send}}(1^\lambda, v, m, r^{\mathsf{send}}) \to c_2$ is a deterministic algorithm that takes as input the security parameter 1^λ, the message v, the receiver's first-round message m, and the randomness r^{send} used to generate the first round message c, and produces the sender's second message c_2.
- $\mathsf{NMC}_{\mathsf{verify}}(1^\lambda, \tau, r^{\mathsf{recv}}) \to 0$ or 1 is a deterministic algorithm that takes as input a transcript $\tau = (c, m, c_2)$ along with the receiver's randomness r^{recv} and outputs 1 if and only if the the transcript is a valid NMC transcript.

In addition, we assume there exists an (unbounded-time) algorithm $\mathsf{NMC}_{\mathsf{extract}}(\tau)$ which takes as input a transcript of the commitment scheme and outputs a value v or $\perp$, such that no opening exists for any value other than v. Thus $\mathsf{NMC}_{\mathsf{extract}}$ defines the unique value committed to by τ.

The definition in this section is taken with small modifications from [51].

We follow the definition of non-malleable commitments introduced by Pass and Rosen [66] and further refined by Lin et al. [55] and Goyal [41] (which in turn build on the original definition of [31]). In the real interaction, there is a man-in-the-middle adversary MIM interacting with a committer $\mathcal{C}$ (where $\mathcal{C}$ commits to value v) in the left session, and interacting with receiver $\mathcal{R}$ in the right session. Prior to the interaction, the value $v \in \{0,1\}^n$ is given to C as local input. MIM receives an auxiliary input z, which might contain a-priori information about v. Then the commit phase is executed. Let $\mathsf{MIM}_{\langle \mathcal{C},\mathcal{R}\rangle}(1^\lambda, v, z)$ denote a random variable that describes the value $\tilde{v}$ committed by the MIM in the right session, jointly with the view of the MIM in the full experiment. In the simulated experiment, a PPT simulator Sim directly interacts with the $\mathcal{R}$. Let $\mathsf{Sim}_{\langle C,R\rangle}(1^\lambda, n, z)$ denote the random variable describing the value $\tilde{v}$ committed to by Sim and the output view of Sim. If the tags in the left and right interaction are equal, the value $\tilde{v}$ committed in the right interaction is defined to be $\perp$ in both experiments.

Concurrent non-malleable commitment schemes consider a setting where the MIM interacts with committers in polynomially many (a-priori unbounded) left sessions, and interacts with receiver(s) in up to $\ell(n)$ right sessons. If any of the tags used by MIM (in any right session) are equal to any of the tags in any left session, we set the value committed by the MIM to be $\perp$ for that session. Then we let $\mathsf{MIM}_{\langle \mathcal{C},\mathcal{R}\rangle}(1^\lambda, v, z)^{\mathsf{many}}$ denote the joint distribution of all the values committed to by the MIM in all right sessions, together with the view of the MIM in the full experiment, and $\mathsf{Sim}_{\langle \mathcal{C},\mathcal{R}\rangle}(1^\lambda, n, z)^{\mathsf{many}}$ denote the joint distribution of all values committed to by the simulator Sim (with access to the MIM) in all right sessions together with the simulated view of MIM.

Definition 8 ((T,ϵ)-Non-Malleable Commitments w.r.t. Commitment.). *A commitment scheme $\langle \mathcal{C}, \mathcal{R}\rangle$ is said to be (T,ϵ)-non-malleable if for every probabilistic time-T MIM there exists a time-$\mathsf{poly}(T)$ simulator Sim such that for every ensemble $\{(v_\lambda, z_\lambda)\}_\lambda$ of polynomial-length strings v_λ and z_λ, the following ensembles are ϵ-indistinguishable by any time-$T(\lambda)$ adversary:*

$$\{\mathsf{MIM}_{\langle \mathcal{C},\mathcal{R}\rangle}(v_\lambda, z_\lambda)\}_\lambda \text{ and } \{\mathsf{Sim}_{\langle C,R\rangle}(1^\lambda, |v_\lambda|, z_\lambda)\}_\lambda$$

Definition 9 ((ℓ,T,ϵ)-Concurrent Non-Malleable Commitments w.r.t. Commitment.). *A commitment scheme $\langle \mathcal{C}, \mathcal{R}\rangle$ is said to be (ℓ,T,ϵ)-concurrent non-malleable if for every probabilistic time-T MIM there exists a time-$\mathsf{poly}(T)$ simulator Sim such that for every ensemble $\{(v_\lambda, z_\lambda)\}_\lambda$ of polynomial-length strings v_λ and z_λ, the following ensembles are ϵ-indistinguishable by any time-$T(\lambda)$ adversary:*

$$\{\mathsf{MIM}_{\langle \mathcal{C},\mathcal{R}\rangle}(v_\lambda, z_\lambda)^{\mathsf{many}}\}_\lambda \text{ and } \{\mathsf{Sim}_{\langle \mathcal{C},\mathcal{R}\rangle}(1^\lambda, |v_\lambda|, z_\lambda)^{\mathsf{many}}\}_\lambda$$

We say that a commitment scheme is fully concurrent with respect to commitment if it is concurrent for any a-priori unbounded polynomial $\ell(n)$.

In our construction, we will require that the particular two-round simultaneous-message non-malleable commitment we use is binding with respect to the first

round, meaning that it is impossible to open any transcript starting with $\mathsf{NMC}_1^{\mathsf{send}}(1^n, v)$ to any value other than v. This is easy to achieve; in particular, we can modify the non-malleable commitment scheme so that the committer additionally sends a non-interactive perfectly-binding commitment to v in the first round, and includes the opening to this commitment as part of its message which it commits to in the non-malleable commitment.

B Two-Round PAKE in the Plain Model

We show how our concurrent two-party computation in Section 3.2 enables us to construct Password-Authenticated Key-Exchange (PAKE) in the plain model. Briefly in a PAKE protocol, each user $\mathsf{U}_{i\in\{1,2\}}$ owns a password $\mathbf{x}_i$ in a certain language $\mathcal{L}_i$. If the two passwords match ($\mathbf{x}_1 = \mathbf{x}_2$), then the users must learn a common high-entropy secret, otherwise they learn nothing about the other user's values.

We observe that our concurrent 2PC in in Section 3.2 directly yields a two-round concurrently secure PAKE construction in the plain model. More precisely, let each party P_i in the concurrent 2PC in Section 3.2 plays the role as the user U (described above) for a functionality f defined as follows:

- Inputs: $(\mathbf{x}_1, k_1)$ from P_1 and $(\mathbf{x}_2, k_2)$ from P_2.
- Check if $\mathbf{x}_1 = \mathbf{x}_2$ and return $k_1 \oplus k_2$ if this is the case.
- Otherwise output $\perp$.

The parties can then feed as input to this functionality their passwords $(\mathbf{x}_1, \mathbf{x}_2)$ along with two keys (k_1, k_2) which are uniformly and freshly sampled for each run of the protocol. Next, we argue that this simple protocol achieves the standard security of PAKE.

Security of PAKEs. In the cryptography literature there are the following two leading paradigms for rigorously defining the above intuition: (i) *Simulation-based security*, in which it works in the framework of universal composability (UC) [25]. This approach works by first defining an appropriate ideal functionality for PAKE; a PAKE protocol is then considered secure if it realizes that functionality. Canetti *et al.* [26] pursued this approach, and defined a PAKE functionality that explicitly allows an adversary to make password guesses;in their work, a random session key is generated unless the adversary's password guess is correct. (ii) *Game-based security,* that is introduced by Bellare-Pointcheval-Rogaway (BPR) [16], which we will proof our PAKE construction in this model. In the following we recall the BPR model in more details.

Game-based security. In the game-based security (the BPR model) [16] definition, a password is chosen from a distribution with min-entropy k, and the security experiment considers interactions of an adversary with multiple instances of the PAKE protocol using that password. A PAKE protocol is considered secure if no probabilistic polynomial time adversary can distinguish a real session key

from a random session key with an advantage better than $Q \cdot 2^k + \mathsf{negl}(\lambda)$, where Q is the number of online attacks by the adversary i.e., actively interfering in Q sessions of the protocol and can make at most Q password guesses.

We briefly recall the formal security model for password key exchange protocols as presented in Bellare et al. [16]. The text here is lifted almost verbatim from [40].

- *Initialization phase.* Before any execution of the protocol, there is an initialization phase during which public parameters are established. We assume a fixed set User of all the users participating in the protocol. For every distinct $U, \hat{U} \in \mathsf{User}$, we assume U and $\hat{U}$ share a password $\mathbf{x} := \mathsf{pw}$. We make the simplifying assumption that each pw is chosen independently and uniformly at random from the set $\{1, \cdots, D_\lambda\}$ for some integer D_λ that may depend on λ.
- *Execution of the protocol.* In the real world, a protocol determines how principals behave in response to input from their environment. In the formal model, these inputs are provided by the adversary $\mathcal{A}$. Each principal can execute the protocol multiple times (possibly concurrently) with different partners; this is modeled by allowing each principal to have an unlimited number of instances with which to execute the protocol. We denote instance i of user U as $\mathsf{instance}_{U,i}$. Each instance may be used only once. The adversary is given oracle access to these different instances; furthermore, each instance maintains (local) state which is updated during the course of the experiment. In particular, each instance $\mathsf{instance}_{U,i}$ maintains local state that includes the following variables:
  - $\mathsf{sid}_{U,i}$, $\mathsf{pid}_{U,i}$, and $\mathsf{sk}_{U,i}$ denote the session id, partner id, and session key, respectively. The session id is simply a way to keep track of different executions; we let $\mathsf{sid}_{U,i}$ be the (ordered) concatenation of all messages sent and received by $\mathsf{instance}_{U,i}$. The partner id denotes the user with whom $\mathsf{instance}_{U,i}$ believes it is interacting; we require $\mathsf{pid}_{U,i} \neq U$.
  - $\mathsf{acc}_{U,i}$ and $\mathsf{term}_{U,i}$ are flags denoting acceptance and termination, respectively.

The adversary's interaction with various instances is modeled via access to the following oracles:

- $\mathsf{Send}(U, i, M)$. This sends message M to instance $\mathsf{instance}_{U,i}$. This instance runs according to the protocol specification, updating state as appropriate. The output of $\mathsf{instance}_{U,i}$ (i.e., the message sent by the instance) is given to the adversary.
- $\mathsf{Execute}(U, i, \hat{U}, j)$. If $\mathsf{instance}_{U,i}$ and $\mathsf{instance}_{\hat{U},j}$ have not yet been used, this oracle executes the protocol between these instances and gives the resulting transcript to the adversary. This models passive eavesdropping of a protocol execution.
- $\mathsf{Reveal}(U, i)$. This outputs the session key $\mathsf{sk}_{U,i}$, modeling leakage of session keys due to, e.g., improper erasure of session keys after use, compromise of a host computer, or cryptanalysis.

– $\mathsf{Test}(U, i)$. This oracle does not model any real-world capability of the adversary, but is instead used to define security. A random bit b is chosen; if $b = 1$ the adversary is given $\mathsf{sk}_{U,i}$, and if $b = 0$ the adversary is given a session key chosen uniformly from the appropriate space.

The security of key exchange protocols is composed of two components: correctness and privacy.
Partnering. Let U, $\hat{U} \in \mathsf{User}$, where User is the set of the users. Instances $\mathsf{instance}_{U,i}$ and $\mathsf{instance}_{\hat{U},j}$ are partnered if: (i) $\mathsf{sk}_{U,i} = \mathsf{sk}_{\hat{U},j} \neq \mathsf{Null}$; and (ii) $\mathsf{pid}_{U,i} = \hat{U}$ and $\mathsf{pid}_{\hat{U},j} = U$.
Correctness. To be viable, a PAKE protocol must satisfy the following notion of correctness: if $\mathsf{instance}_{U,i}$ and $\mathsf{instance}_{\hat{U},j}$ are partnered then $\mathsf{acc}_{U,i} = \mathsf{acc}_{\hat{U},j} = \mathsf{true}$ and $\mathsf{sk}_{U,i} = \mathsf{sk}_{\hat{U},j}$, i.e., they both accept and conclude with the same session key.
Privacy. Intuitively, a protocol achieves privacy if the adversary cannot distinguish real session keys from random ones. Formally, we say that the adversary succeeds if it correctly guesses the bit determining whether it received the real session key or a random session key in the Test oracle query. Of course, the adversary can always correctly guess the bit in a $\mathsf{Test}(U, i)$ query if it queried $\mathsf{Reveal}(U, i)$ when U and $\hat{U}$ are partnered. Therefore, $\mathcal{A}$ is only said to have succeeded if these oracles were not queried. Now, the adversary's advantage is formally defined by:

$$\mathsf{adv}_{\mathcal{A},i}(\lambda) = |2 \cdot \Pr[\mathcal{A} \text{ succeeds}] - 1|$$

We reiterate that an adversary is only considered to have succeeded if it correctly guesses the bit used by the $\mathsf{Test}(U, i)$ oracle and it did not query $\mathsf{Reveal}(U, i)$ or $\mathsf{Reveal}(\hat{U}, J)$ when U and $\hat{U}$ are partnered.

Security of our concurrent-secure PAKE. Now, in Theorem 4 we analyze security of our concurrent-secure PAKE in the BPR [16] security model with super-polynomial simulation.

Theorem 4. *Let the concurrent-secure 2PC in Section 3.2 be secure against malicious adversaries in the plain model. Then the PAKE protocol* `PAKE` *based on the concurrent-secure 2PC is complete and private in the plain model.*

Proof. **(i: Completeness).** This is straightforward from the construction and follows directly from the correctness of the concurrent 2PC in Section 3.2.

(ii: Privacy). We now proceed to prove the privacy requirement through a series of hybrid experiments. Fix a PPT adversary $\mathcal{A}$ attacking the protocol. We use a hybrid argument to bound the advantage of the adversary $\mathcal{A}$. Let hybrid $\mathcal{H}_0$ represent the initial experiment, in which $\mathcal{A}$ interacts with the real protocol. We define a sequence of experiments $\mathcal{H}_1, \cdots$, and denote the advantage of adversary $\mathcal{A}$ in experiment $\mathcal{H}_i$ as:

$$\mathsf{adv}_{\mathcal{A},i}(\lambda) = |2 \cdot \Pr[\mathcal{A} \text{ succeeds}] - 1|.$$

We bound the difference between the adversary's advantage in successive experiments, and then bound the adversary's advantage in the final experiment; this gives the desired bound on $\mathsf{adv}_0(\lambda)$, the adversary's advantage when attacking the real protocol. We recall that the adversary can only statistically corrupt users, i.e., it can corrupt a password or password hash when no instance of the associated players is involved in an execution of the protocol. Let $\mathtt{c2PC}.M_1$ and Let $\mathtt{c2PC}.M_2$ be the first and the second message of the $\mathtt{c2PC}$ scheme in Section 3.2, respectively. We separate Send queries in three types:

- $\mathsf{Send}_1(P_1, \mathsf{start}, P_2)$ queries which enable an adversary to ask a client P_1 to initiate the protocol with a server P_2 and which return the first flow for P_1 and P_2.
- $\mathsf{Send}_2(P_1, P_2, M)$ queries which enable an adversary to send the first flow for P_1 and P_2 and which return the second flow answered back by P_2.
- $\mathsf{Send}_3(P_1, P_2, M)$ queries which enable an adversary to send the second flow for P_1 and P_2 and which return nothing but set the session key f of P_2.

Hybrid $\mathcal{H}_1$: In this hybrid, we modify the way Execute queries are handled. Namely, in response to a query $\mathsf{Execute}$ we act in the same way as in $\mathcal{H}_0$ except that, it runs the simulator of the $\mathtt{c2PC}$ and simulates the protocol. Assuming the security (the privacy property) of the $\mathtt{c2PC}$ of the proof of theorem 2, $\mathcal{H}_0$ and $\mathcal{H}_1$ are computationally indistinguishable. Thus, $|\mathsf{adv}_{\mathcal{A},\mathcal{H}_1}(\lambda) - \mathsf{adv}_{\mathcal{A},\mathcal{H}_0}(\lambda)| \leq \mathsf{negl}(\lambda)$.

Hybrid $\mathcal{H}_2$: In this experiment we begin to modify the Send oracle. We now, in response to the query $\mathsf{Send}_2(P_1, P_2, M)$ (when a second flow is sent, in the name of some client instance P_1 and to some server instance P_2) we proceed as follows: if the password of P_2 is corrupted, we answer honestly and compute the session key honestly. Otherwise, three cases can appear:

- if $\mathsf{pid}_{P_1,i} \neq P_2$ then aborts $\mathsf{instance}_{P_1,i}$ as it would in $\mathcal{H}_1$. From here on we assume this is not the case.
- If M is not *previously-used*, then we run the extraction phase of $\mathtt{c2PC}$ on P_1, the receiver message, to obtain the commit ed values) and if they are well-formed, the adversary is declared successful and the experiment ends (in other words we choose the session key $f = \perp$; if later the adversary asks the session key via a Test-query, we stop the simulation and let it win). Otherwise, we choose the session key f at random;
- If the second flow M was output by a previous query Send_1 then we say that the message M is *previously-used* and the experiment continues as in the last $\mathcal{H}$. More precisely, if it is a replay of a previous flow sent by the simulator, then, in particular, we know the secret values associated with M, and we compute the session key f using them.

The change in the first case can only increase the advantage of the adversary, while the change second change is indistinguishable under the privacy reservation of $\mathtt{c2PC}$ protocol and thus only increases the advantage of the adversary by a negligible term. The change in the last case does not change the advantage of the adversary. Therefore, we have: $\mathsf{adv}_{\mathcal{A},\mathcal{H}_1}(\lambda) \leq \mathsf{adv}_{\mathcal{A},\mathcal{H}_2}(\lambda) + \mathsf{negl}(\lambda)$.

Hybrid $\mathcal{H}_3$: Now, we modify the way of $\mathsf{Send}_3(P_1, P_2, M)$ queries are answered, similarly to what we have done for the queries in $\mathcal{H}_2$. More precisely, when a second flow M is sent, in the name of some server instance P_2 and to some client instance P_1, if the password of P_1 has been corrupted, we answer honestly and compute the session key honestly. Otherwise, this case can appear that the message is not a message generated by P_2 (via a Send_2 query) after receiving the first flow sent by P_1 (via a Send_1 query), then we first run the extraction phase of $\mathtt{c2PC}$ on P_2 to obtain the commit ed values) and if they are well-formed, the adversary is declared successful and the experiment ends (in other words we choose the session key $f = \perp$; if later the adversary ask the session key via a Test-query, we stop the simulation and let it win). Otherwise, we choose the session key f at random.
The change in the first case can only increase the advantage of the adversary, while the changes in the later case are indistinguishable under privacy reservation of $\mathtt{c2PC}$ scheme, and thus only increase the advantage of the adversary by a negligible term. Thus, $\mathsf{adv}_{\mathcal{A},\mathcal{H}_2}(\lambda) \leq \mathsf{adv}_{\mathcal{A},\mathcal{H}_3}(\lambda) + \mathsf{negl}(\lambda)$.

Hybrid $\mathcal{H}_4$: In this experiment we modify the $\mathsf{Send}_1(P_1, \mathsf{start}, P_2)$ oracle. In response to a Send_1 query, when the password of the client P_1 is not corrupted, we send the simulated $\mathtt{c2PC}.M_2$ message.
This is indistinguishable under the chooser's security of $\mathtt{c2PC}$ scheme, thus, $|\mathsf{adv}_{\mathcal{A},\mathcal{H}_4}(\lambda) - \mathsf{adv}_{\mathcal{A},\mathcal{H}_3}(\lambda)| \leq \mathsf{negl}(\lambda)$.

Hybrid $\mathcal{H}_5$: We now modify the way we answer to Send_2 queries for replayed messages. When a first message is replayed by the adversary, we choose the session key at random (except when the password of client P_1 was corrupted, in which case the flow is honestly generated and the session key too). This is indistinguishable, thanks to privacy preservation of $\mathtt{c2PC}$ scheme: $|\mathsf{adv}_{\mathcal{A},\mathcal{H}_5}(\lambda) - \mathsf{adv}_{\mathcal{A},\mathcal{H}_4}(\lambda)| \leq \mathsf{negl}(\lambda)$.
We remark that, in this game, all session keys returned by the Test queries are completely independent and random.
In this last game, the simulator does not use the passwords, and so no information leaks except in case of corruption. We also remark that the adversary wins only:

- if it sends a flow to a server P_2 (with non corrupted password), from which the simulator ($\mathtt{c2PC}$'s simulator) extracts a valid secret values of the client P_1's message, such that they are well-formed; and if the adversary makes a Test-query for that session,
- or if it sends a flow to a client P_1, from which the simulator extracts a valid secret values of the server P_2's message, such that they are well-formed; and if the adversary makes a Test-query for that session.

If passwords are chosen independently at random from a distribution D of min-entropy β (i.e., not in the related- password model), and if no corruption occurs, then, for each flow, the events happen with probability at most $2^{-\beta} + \mathsf{negl}(\lambda)$ thanks to the entropy preservation.
So finally, with no corruption and independently chosen passwords, the probability of the adversary to win this last game (and also the original game) is

at most $Q_s \cdot 2^{-\beta} + \mathsf{negl}(\lambda)$, where Q_s is the number of active sessions (sessions for which the adversary has sent a non-honestly-generated flow).

□