Two-Round Concurrently Secure Two-Party Computation

Behzad Abdolmaleki Giulio Malavolta Ahmadreza Rahimi

Max Planck Institute for Security and Privacy

Abstract. In this paper, we study the round complexity of concurrently secure computation protocols in the plain model, without random oracles or assuming the presence of a trusted setup. In the plain model, it is well known that concurrently secure two-party computation with polynomial simulation is impossible to achieve in two rounds. For this reason, we focus on the well-studied notion of security with super-polynomial simulation (SPS). Our main result is the first construction of two-round SPS two-party computation for general functionalities in the plain model. Prior to our work, we only knew three-round constructions [Badrinarayanan et al., TCC 2017] and two-round protocols were not known from any computational assumption.

As immediate applications, we establish the feasibility result for a series of cryptographic primitives of interest, such as: Two-round password authentication key exchange (PAKE) in the plain model, two-round concurrent blind signature in the plain model, and two round concurrent computation for quantum circuits (2PQC).

1 Introduction

Secure computation protocols enable mutually distrustful parties to compute a functionality without compromising the privacy of their inputs. Secure computation protocols [GMW87,Yao86] are a conrnerstone of modern cryptography and have been studied in a variety of settings. In the specific setting of two-party computation (2PC), two parties with inputs $\mathtt{x}_0$ and $\mathtt{x}_1$, wish to jointly compute a functionality $\mathtt{f}(\mathtt{x}_0, \mathtt{x}_1)$. Loosely speaking, the security requirements are that nothing can be learned from the protocol other than the output (privacy), and that the output is distributed according to the prescribed functionality (correctness). These security requirements must hold in the face of an adversary who controls one of the parties and can arbitrarily deviate from the protocol instructions (i.e., in this work we consider malicious adversaries). The classical notion of security only considers the stand-alone setting, where security holds only if a single protocol session is executed in isolation. As it has become increasingly evident over the last decades, stand-alone security does not suffice in real-world scenarios where several protocol sessions may be executed concurrently (a typical example being protocols executed over modern networked environments, such as the Internet).

The Concurrent Setting. A more general (and realistic) setting considers the case where many protocol executions are run concurrently within a network. Unfortunately, the security of a protocol in the stand-alone setting does not necessarily imply its security under concurrent composition [FS90]. Secure computation in the concurrent setting is more challenging to define than the stand-alone setting. In the literature, the holy grail of concurrent security is the notion of Universal Composability (UC) Canetti [Can01] which, among other many benefits, allows for a modular design and analysis of protocols. For each cryptographic task, an ideal functionality can be defined, which incorporates the required properties of a protocol for the task and the allowed actions of an adversary. A protocol is said to securely realize the ideal functionality if, loosely speaking, any effect caused by an adversary attacking the protocol can be obtained by an adversary attacking the ideal functionality.

UC security, however, turns out to be too strong to achieve in the plain model (i.e., without any trusted setup or random oracle). In particular, it has been shown that UC-secure 2PC is impossible to achieve in the plain model [Can01,CKL03]. This result has been extended by Lindell [Lin08], who show that even concurrently secure 2PC is impossible if simulation-based definitions of security are used. He proved that even in the particular case where only instantiations of the same protocol are allowed, the standard notion of polynomial time simulation is impossible to achieve.

In an effort to overcome the above mentioned impossibility results, many recent works have focused on proving concurrent security for 2PC in alternative models, e.g., in the bounded concurrent model [Pas04a], in the multiple ideal-query model [GJ13], and for input-indistinguishable computation [MPR06]. A common relaxation of polynomial-time simulation is the notion of security with super-polynomial simulators (SPS) [Pas03b,PS04,BS05]. 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 [MPR06]. There has been a fruitful line of research devoted to understanding the power of SPS security for secure computations in the concurrent setting [MMY06,CLP10,GGJS12,LP12,PLV12,KMO14,Kiy14,GLP+15,GKP17,BGJ+17].

The Round Complexity of 2PC. One of the most meaningful performance measures for a 2PC is its round complexity. A series of breakthrough results has established that two rounds are both necessary and sufficient for stand-alone 2PC [GS17,GS18,BL18]. In the concurrent setting, a series of works [GGJS12,KMO14] constructed constant-round protocols (approximately 20 rounds) in the simultaneous message exchange model. Later, Garg *et al.* [GKP17] decreased the round complexity to 5 rounds with SPS security from standard sub-exponential assumptions. Recently, Badrinarayanan *et al.* [BGJ+17] constructed a concurrent MPC with SPS security with 3 rounds against Byzantine adversaries, assuming sub-exponentially secure DDH and LWE.[1] This raises the following natural question:

Can we achieve a two-round concurrently secure two-party computation in the plain model?

In this work we give a positive answer to the above question. It is well known that for any notion of simulation-based security two rounds are necessary. Thus our work establishes the feasibility of round-optimal concurrent 2PC with SPS security.

1.1 Our Results

We present a new secure computation protocol whose round complexity matches that of the stand alone setting. More specifically, we present a two-round SPS-secure (simultaneous-round) concurrent 2PC protocol. Our concurrent 2PC construction is based on the existence (all with subexponential security) of non-interactive $\mathtt{CCA}$-commitments and the hardness of the learning with errors (LWE) problem. While the latter is a standard assumption, non-interactive $\mathtt{CCA}$-commitment schemes have been recently shown to exist assuming indistinguishability obfuscation [Khu21], which in turn can be realized from well-founded assumptions [JLS20].

Application: Concurrently Secure PAKE. In a password-authenticated key exchange (PAKE) protocol, two users hold passwords $(\mathtt{x}_1, \mathtt{x}_2)$ and want to exchange a high-entropy secret if $\mathtt{x}_1 = \mathtt{x}_2$, otherwise they learn nothing about the other user's inputs. We show that our concurrently secure 2PC protocol directly provides us a two-round (concurrently secure) PAKE scheme in the plain model. Prior to our work, even two-round PAKE scheme (not concurrent) in the plain model was left as an open problem.

Application: Concurrently Secure Blind Signatures. Blind signatures [Cha82] enable users to obtain a signature without revealing a message to be signed to a signer. More precisely, a blind signature scheme is a 2PC between a signer and a user: The signer holds a key pair, and the user holds a message. At the end of the interaction, the user receives a signature on the message without revealing the message to the signer. Blind signatures are used as a building block for various other privacy-preserving cryptosystems such as e-voting [FOO92,Cha88], anonymous credential [CL01], and direct anonymous attestation [BCC04]. Before our work, constructing a blind signature that satisfies both round-optimal and concurrent security in the plain model was left an open problem. Using our concurrently secure 2PC, we present the first round-optimal

[1] [BGJ+17] also constructed a two-round concurrent MPC with SPS security against Byzantine adversaries but for *input-less randomized functionalities*, assuming sub-exponentially secure indistinguishability obfuscation and DDH.

concurrently-secure blind signature scheme that does not rely on random oracles or any setup assumptions such as a common reference string.

Application: Quantum Computation. As a bonus, we note that our simulator is straight-line and black-box and therefore our 2PC can be plugged into the recent work of Bartusek*et al.* [BCKM20], providing the first quantum 2PC in the plain model.

1.2 Concurrent Work

In a concurrent and independent work, Fernando, Jain, and Komargodski [FJK21] also present new results on two-round concurrent secure computation. They also consider the notion of security with super-polynomial simulation and obtain results assuming the quantum hardness of LWE, the classical hardness of SXDH, the existence of indistinguishability obfuscation and time-lock puzzles. We highlight two main differences of our approaches:

- Our approach is limited to the case of 2 parties, whereas their protocol supports polynomially-many (in fact an unbounded polynomial) parties.
- In terms of assumption, our work uses a strict subset of their set of assumptions, namely, LWE and the existence of indistinguishability obfuscation.[2]
- We also obtain results on concurrent 2PC for quantum functionalities. Due to the reliance of the SXDH assumption, and more broadly on the classical-quantum simulation technique [KK19], their security analysis breaks down in the quantum settings.

1.3 Technical Overview

We now give an overview of the techniques used in our work. We start by describing the difficulty of constructing round-optimal concurrently secure 2PC with SPS security of general functionalities with super-polynomial simulation (which both parties receive the output of the computation). After that, we give a brief intuition on the main ideas for overcoming such barriers. For a more derailed exposition of the protocols, we refer the reader to the technical sections.

Challenges in the Concurrent Setting. As the fist step of constructing our round-optimal concurrent 2PC, we use a maliciously circuit-private fully homomorphic encryption (FHE) scheme and let each party encrypts her input $\mathtt{x}$ in the first round of our concurrent 2PC construction. In circuit-private FHE schemes, one party (the receiver) holding an input $\mathtt{x}$ wishes to learn the evaluation of a circuit $\mathcal{C}$ held by another party (the sender). That is, even if both maliciously formed public key and ciphertext are used, encrypted outputs only reveal the evaluation of the circuit on some well-formed input $\mathtt{x}$. One immediate issue with this approach is that parties may compute their encryption by mauling the ciphertext of the other party and introduce correlations between the two inputs, thus violating security. To avoid such attacks, we additionally let each party to commit to her input $\mathtt{x}$ via a non-interactive CCA-secure commitment scheme[3]. In the second round, we let parties (symmetrically) evaluate the function $\mathtt{f}(x_i, \cdot)$ homomorphically and send over the resulting ciphertext $\hat{\mathtt{c}}_i$, which is an encryption of $\mathtt{f}(\mathtt{x}_0, \mathtt{x}_1)$. Each party can then recover the function output by simply decrypting the incoming ciphertext.

Ensuring Input Consistency. The main challenge of this approach is then to ensure that the same value $\mathtt{x}$ is used in the both in the FHE and in the CCA-secure commitment, for both the first and the second round of interaction. For the second round we can use 2-round SPS zero-knowledge to guarantee this [Pas03a], however for the first round this is not a viable option, since non-interactive zero-knowledge requires a setup. To resolve this, we resort to using conditional disclosure of secrets (CDS). Roughly speaking, this primitive guarantees that, if the non-interactive CCA-secure commitments in the first round are not consistence with

[2] Obfuscation is only needed to instantiate non-interactive CCA secure commitments. Future improvements in the area of CCA-commitments can be used off-the-shelf to improve the set of assumptions needed for our protocol.

[3] In the security proof, the simulator will use the CCA-secure commitment to extract the input of the adversary.

the FHE ciphertext, then the user will not be able to recover the second message of the protocol. In a sense, we check the well-formedness condition implicitly, by conditioning the delivering of the second round to the receiver having a valid witness that certifies this relation.

A more subtle point is that we need to simulate the zero-knowledge protocol (that guarantees consistency of the second round) while at the same time arguing that the attacker cannot prove false statements. Specifically, when we switch a real ZK proof to being simulated (in super-polynomial time $\mathsf{T}_{\mathsf{Sim}}$), we must argue that the values within CCA commitments provided by the adversary did not suddenly change. To achieve this, it must be true that the quality of the SPS.ZK simulation is sufficiently high to guarantee that the messages inside the CCA commitments did not change. Specifically, we must be able to break the CCA commitments and extract from them in time that is less than the time for which soundness hold T_{zk}. Putting together all these constraints, we have that CCA commitments should be breakable in time that is less than the time against which they remain non-malleable: this is a direct contradiction. Therefore, we need a ZK argument systems that $\mathsf{T}_{\mathsf{Sim}} \ll \mathsf{T}_{\mathsf{zk}}$. To resolve this, we use zero-knowledge with strong simulation. Such a primitive was recently realized by [KS17], by constructing a new form of two-round extractable commitments. Carefully setting the parameters of our scheme, we obtain the desired security. We refer to Section 3 for more details of the construction and the proof's techniques.

2 Preliminaries

We typically use λ to denote the security parameter. A function $\mathsf{negl}(\lambda) : \mathsf{N} \mapsto [0,1]$ is called negligible $\mathsf{negl}(\lambda)(\lambda) = O(\lambda^{-c})$ for every constant $c \in \mathsf{N}$. We denote this by $p \approx 0$. We define by $\mathsf{negl}(\lambda)$ a negligible function of λ. We say that $\mathsf{T}_1(\lambda) \gg \mathsf{T}_2(\lambda)$ if $\mathsf{T}_1(\lambda) > \mathsf{T}_2(\lambda) \cdot \lambda^c$ for all constants c. We define a T-time machine as a non-uniform Turing Machine that runs in time at most T. All honest parties in definitions below are by default uniform interactive Turing Machines, unless otherwise specified.

2.1 CCA-Secure Commitment

Commitment Schemes. A commitment scheme is a two-party protocol (with two phases) in which one party, the sender, commits himself in the first phase (the commit phase) to a value while keeping it secret from the other party, the receiver. In the second phase (the open phase) the sender reveals the value he committed to. At the end of this phase the receiver outputs this value. In addition to the requirement that both sender and receiver run in polynomial time, we require that a commitment scheme satisfies the following two properties: (i) the commit phase yields no knowledge of the value to the receiver (Hiding). (ii) Given the transcript of the interaction in the first phase, there exists at most one value that the receiver can accept as the correct opening in the reveal phase. This also applies to cheating senders (Binding). For a formal definition see [Ode01].

In a tag-based commitment scheme both parties get a bit string called tag as additional input. We denote by $\mathsf{Com}(m)$ a (possibly interactive) commitment to the value $m \in \{0,1\}^\lambda$ under the tag $\mathsf{tag} \in \{0,1\}^\lambda$ using the commitment scheme com.

CCA-secure commitment. Briefly, a tag-based commitment scheme Com is said to be CCA-secure [LP12], if the value committed to using a tag tag remains hidden even if the receiver has access to an oracle that breaks polynomially many commitments using a different tag $\mathsf{tag}' \neq \mathsf{tag}$ for her.

Definition 1 (CCA-secure commitment scheme.). *Let* Com *be a tag-based commitment scheme and* $\mathsf{O}_{\mathtt{CCA}}$ *be the* CCA*-oracle for* Com*. We say that* Com *is* CCA*-secure, if for any* $\lambda \in \mathsf{N}$ *and for every PPT adversary* $\mathcal{A}$*,*

$$\Pr\left[\begin{matrix}(\mathsf{tag}, m_0, m_1) \leftarrow \mathcal{A}^{\mathsf{O}_{\mathtt{CCA}}}(\lambda); b \leftarrow\!\!\$\, \{0,1\}; \mathsf{c}_b \leftarrow \mathsf{Com}(m_b) : \\ b' \leftarrow \mathcal{A}^{\mathsf{O}_{\mathtt{CCA}}}(\mathsf{c}_b) \wedge b = b'\end{matrix}\right] \approx 0 \ .$$

Where CCA*-oracle* $\mathsf{O}_{\mathtt{CCA}}$ *for* Com *acts as follows in an interaction with an adversary* $\mathcal{A}$*: It participates with* $\mathcal{A}$ *in polynomially many sessions of the commit phase of* Com *as an honest receiver (the adversary determines*

the tag he wants to use at the start of each session). At the end of each session, if the session is valid, the oracle returns the unique value m committed to in the interaction; otherwise, it returns $\perp$. Note that if a session has multiple valid committed values, the experiment also returns $\perp$. Note that if during the execution $\mathcal{A}$'s queries the oracle on a commitment that uses the challenge tag tag*, it returns* $\perp$*.*

Instantiations. Non-interactive non-malleable commitments have been extensively studied in the literature and schemes are known from various cryptographic assumptions, such as adaptive one-way functions [PPV08], or (sub-exponentially hard) indistinguishability obfuscation [Khu21][4]. In some cases we will allow for a two-round commitments, with a reusable first round. For such instances, we can also use constructions from sub-exponentially secure time-lock puzzles [LPS17].

2.2 Circuit-Private (Leveled) Fully-Homomorphic Encryption

We recall a leveled fully-homomorphic encryption scheme with circuit privacy, that is, for an encryption $\mathsf{c} = \mathtt{FHE.Enc}(\mathsf{x})$ and a circuit $\mathcal{C}$, a $\mathcal{C}$-homomorphically-evaluated ciphertext $\mathsf{c} = \mathtt{FHE.Eval}(\mathcal{C}, \mathsf{c})$ reveals nothing on $\mathcal{C}$ but $\mathcal{C}(\mathsf{x})$ [OPP14].

Definition 2 (Circuit-private fully-homomorphic encryption.). *A (maliciously) circuit-private,leveled fully-homomoprhic encryption (*$\mathtt{FHE}$*) scheme*
$(\mathtt{FHE.Gen}, \mathtt{FHE.Enc}, \mathtt{FHE.Eval}, \mathtt{FHE.Dec})$ *has the following syntax:*

- $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathtt{FHE.Gen}(1^\lambda, 1^{s(\lambda)})$*: a probabilistic algorithm that takes a security parameter* λ *and a circuit size bound* $s(\lambda)$ *and outputs a secret key* sk *and its corresponding public key* pk*.*
- $\mathsf{c} \leftarrow \mathtt{FHE.Enc}(\mathsf{pk}, \mathsf{x})$*: a probabilistic algorithm that given* pk *and a string* $\mathsf{x} \in \{0,1\}^*$*, outputs a ciphertext* c*.*
- $\hat{\mathsf{c}} \leftarrow \mathtt{FHE.Eval}(\mathsf{pk}, \mathcal{C}, \mathsf{c})$*: a probabilistic algorithm that takes a* pk*, a circuit* $\mathcal{C}$ *and a ciphertext* c*, and outputs an evaluated ciphertext* $\hat{\mathsf{c}}$*.*
- $\hat{\mathsf{x}} \leftarrow \mathtt{FHE.Dec}(\mathsf{sk}, \hat{\mathsf{c}})$*: a deterministic algorithm that takes a* sk *and a ciphertext* $\hat{\mathsf{c}}$ *and outputs a string* $\hat{\mathsf{x}} := \mathcal{C}(\mathsf{x})$*.*

The scheme satisfies the following,

- **Perfect correctness.** For any polynomial $s(\cdot)$, for any $\lambda \in \mathsf{N}$, size-$s(\lambda)$ classical circuit C and input x for C,
$$\Pr\left[\begin{matrix}(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathtt{FHE.Gen}(1^\lambda, 1^{s(\lambda)}); \mathsf{c} \leftarrow \mathtt{FHE.Enc}(\mathsf{pk}, \mathsf{x}); \\ \hat{\mathsf{c}} \leftarrow \mathtt{FHE.Eval}(\mathsf{pk}, \mathcal{C}, \mathsf{c}) : \mathtt{FHE.Dec}(\mathsf{sk}, \hat{\mathsf{c}}) = \hat{\mathsf{x}}\end{matrix}\right] = 0 \ .$$
- **Input privacy.** For every polynomial $l(\cdot)$ (and any polynomial $s(\lambda)$),
$$\{\, \mathsf{c} : \begin{matrix}(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathtt{FHE.Gen}(1^\lambda, 1^{s(\lambda)}); \\ \mathsf{c} \leftarrow \mathtt{FHE.Enc}(\mathsf{pk}, \mathsf{x}_0)\end{matrix} \,\} \approx \{\, \mathsf{c} : \begin{matrix}(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathtt{FHE.Gen}(1^\lambda, 1^{s(\lambda)}); \\ \mathsf{c} \leftarrow \mathtt{FHE.Enc}(\mathsf{pk}, \mathsf{x}_1)\end{matrix} \,\}$$
where $\lambda \in \mathsf{N}$ and $\mathsf{x}_0, \mathsf{x}_1 \in \{0,1\}^{l(\lambda)}$.
- **Statistical circuit privacy.** There exist unbounded algorithms, probabilistic $\mathtt{FHE.Sim}$ and deterministic $\mathtt{FHE.Ext}$ such that:
  - For every $\mathsf{x} \in \{0,1\}^*$, $\mathsf{c} \in \mathtt{FHE.Enc}(\mathsf{x})$, the extractor outputs $\mathtt{FHE.Ext}(\mathsf{c}) = x$.
  - For any polynomial $s(\cdot)$,
$$\{\, \mathtt{FHE.Eval}(\mathsf{pk}, \mathcal{C}, \mathsf{c}^*) \,\} \approx \{\, \mathtt{FHE.Sim}(1^\lambda, \mathcal{C}(\mathtt{FHE.Ext}(1^\lambda, \mathsf{c}^*)) \,\},$$
  where $\lambda \in \mathsf{N}$, $\mathcal{C}$ is a $s(\lambda)$-size circuit, and $\mathsf{c}^* \in \{0,1\}^*$.

Instantiations. Circuit-private leveled FHE schemes are known based on LWE [OPCPC14,BD18].

[4] One non-standard aspect of such a scheme is that the commitment transcript is determined also by the receiver's randomness. Looking ahead to our scheme, we will want to prove to the receiver that the commitment is well-formed, which may not be a well-defined statement. However, this can be easily resolved by observing that the well-formedness of the commitment only depends on the committer randomness and thus it suffices to show that the sender's message is computed honestly.

2.3 ZK With Super-polynomial Simulation

We recall the definitions of two-message ZK arguments with super-polynomial simulation (SPS) [Pas04b] and with super-polynomial strong simulation (SPSS)[KS17].

Definition 3 (Two message $(\mathsf{T}_{\mathsf{Sim}}, \mathsf{T}_{zk}, \delta_{zk})$-ZK arguments with super-polynomial simulation). *An interactive proof (or argument) $< \mathsf{P}, \mathsf{V} >$ for the language $\mathcal{L} \in \mathsf{NP}$, with the witness relation $\mathcal{R}_{\mathcal{L}}$, is $(\mathsf{T}_{\mathsf{Sim}}, \mathsf{T}_{zk}, \delta_{zk})$-simulatable if for every T_{zk}-time machine V^* exists a probabilistic simulator Sim with running time bounded by $\mathsf{T}_{\mathsf{Sim}}$ such that the following two ensembles are $(\mathsf{T}_{zk}, \delta_{zk})$-computationally indistinguishable (when the distinguishing gap is a function in $\lambda = |\mathtt{x}|$):*

- $\{< \mathsf{P}(y), \mathsf{V}^*(z) > (\mathtt{x})\}_{z \in \{0,1\}^*, \mathtt{x} \in \mathcal{L}}$ *for arbitrary* $y \in \mathcal{R}_{\mathcal{L}}(\mathtt{x})$.
- $\{\mathsf{Sim}(\mathtt{x}, z)\}_{z \in \{0,1\}^*, \mathtt{x} \in \mathcal{L}}$.

That is, for every probabilistic algorithm $\mathcal{D}$ running in time polynomial in the length of its first input, every polynomial p, all sufficiently long $\mathtt{x} \in \mathcal{L}$, all $y \in \mathcal{R}_{\mathcal{L}}(x)$ and all auxiliary inputs $z \in \{0,1\}^*$ it holds that

$$\Pr[\mathcal{D}(\mathtt{x}, z, < \mathsf{P}(y), \mathsf{V}^*(z) > (\mathtt{x})) = 1] - \Pr[\mathcal{D}(\mathtt{x}, z, \mathsf{Sim}(\mathtt{x}, z)(\mathtt{x})) = 1] \leq \delta_{zk}(\lambda).$$

2.4 ZK with Super-polynomial Strong Simulation

We recall the definition of zero-knowledge with strong simulation from [KS17].

Definition 4 ($(\mathsf{T}_{\Pi}, \mathsf{T}_{\mathsf{Sim}}, \mathsf{T}_{\mathsf{zk}}, \mathsf{T}_{\mathcal{L}}, \delta_{\mathsf{zk}})$-ZK arguments with super-polynomial strong simulation). *An interactive protocol between a PPT prover P with input $(\mathtt{x}, \mathtt{w}) \in \mathcal{R}_{\mathcal{L}}$ for some language $\mathcal{L}$, and PPT verifier V with input $\mathtt{x}$, denoted by $< \mathsf{P}, \mathsf{V} > (\mathtt{x}, \mathtt{w})$, a super-polynomial strong simulation (SPSS) zero-knowledge argument if it satisfies the following properties and $\mathsf{T}_{\Pi} \ll \mathsf{T}_{\mathsf{Sim}} \ll \mathsf{T}_{zk} \ll \mathsf{T}_{\mathcal{L}}$:*

- ***Completeness.*** *For every $(\mathtt{x}, \mathtt{w}) \in \mathcal{R}_{\mathcal{L}}$, $\Pr[\mathsf{V}$ outputs $1 \mid < \mathsf{P}, \mathsf{V} > (\mathtt{x}, \mathtt{w})] \geq 1 - \mathsf{negl}(\lambda)$, where the probability is over the random coins of P and V.*

- T_{Π}-***Adaptive-soundness.*** *For any language $\mathcal{L}$ that can be decided in time at most $\mathsf{T}_{\mathcal{L}}$, every $\mathtt{x}$, every $z \in \{0,1\}^*$, and every poly-non-uniform prover P^* running in time at most T_{Π} that chooses $\mathtt{x}$ adaptively after observing verifier message, $\Pr[< \mathsf{P}^*(z), \mathsf{V} > (\mathtt{x}) = 1 \wedge \mathtt{x} \notin \mathcal{L}] \leq \mathsf{negl}(\lambda)$, where the probability is over the random coins of V.*

- $(\mathsf{T}_{\mathsf{Sim}}, \mathsf{T}_{\mathsf{zk}}, \delta_{\mathsf{zk}})$-***Zero knowledge.*** *There exists a (uniform) simulator Sim that runs in time $\mathsf{T}_{\mathsf{Sim}}$, such that for every $\mathtt{x}$, every non-uniform T_{zk}-verifier V^* with advice z, and every T_{zk}- distinguisher $\mathcal{D}$:*

$$|\Pr[\mathcal{D}(\mathtt{x}, z, \mathsf{view}_{\mathsf{V}^*}[< \mathsf{P}, \mathsf{V}^*(z) > (\mathtt{x}, \mathtt{w})]) = 1] - \Pr[\mathcal{D}(\mathtt{x}, z, \mathsf{Sim}^{\mathsf{V}^*}(\mathtt{x}, z)(\mathtt{x})) = 1]| \leq \delta_{\mathsf{zk}}(\lambda).$$

Notice that, [KS17] presented a construction of an SPSS.ZK scheme satisfying SPSS.ZK these properties that can be based on one of the following sub-exponential assumptions: 1) DDH, 2) Quadratic Residuosity, 3) Nth Residuosity, and 4) LWE [BD18,DGI+19].

2.5 Conditional Disclosure of Secrets

Conditional disclosure of secrets for an NP language $\mathcal{L}$ [AIR01,BP12,AJ17] can be viewed as a two message analog of witness encryption. That is, the sender holds an instance $\mathtt{x}$ and message m and the receiver holds $\mathtt{x}$ and a corresponding witness $\mathtt{w}$. If the witness is valid, then the receiver obtains m, whereas if $\mathtt{x} \notin \mathcal{L}$, m remains hidden. We further require that the protocol hides the witness $\mathtt{w}$ from the sender.

Definition 5 (Conditional disclosure of secrets.). *A conditional disclosure of secrets scheme $\mathtt{CDS} = (\mathtt{CDS.R}, \mathtt{CDS.S}, \mathtt{CDS.D})$ for a language $\mathcal{L} \in \mathsf{NP}$ satisfies:*

– ***Correctness.*** *For any* $(\mathtt{x}, \mathtt{w}) \in \mathcal{R}_{\mathcal{L}}$*, and message* $m \in \{0,1\}^*$*,*

$$\Pr[\mathtt{CDS}.\mathsf{D}_k(\mathtt{ct}_\mathsf{S}) = m | (\mathtt{ct}_\mathsf{R}, k) \leftarrow \mathtt{CDS}.\mathsf{R}(\mathtt{x}, \mathtt{w}); \mathtt{ct}_\mathsf{S} \leftarrow \mathtt{CDS}.\mathsf{S}(\mathtt{x}, m, \mathtt{ct}_\mathsf{R})] = 1.$$

– ***Message indistinguishability.*** *For any polynomial-size distinguisher* $\mathcal{D}$*, there exists a negligible* ϵ *such that for any security parameter* $\lambda \in \mathsf{N}$*, any* $\mathtt{x} \in \{0,1\}^\lambda \notin \mathcal{L}$*,* $\mathtt{ct}^*_\mathsf{R}$ *and two equal-length messages* m_0 *and* m_1*,*

$$\mathtt{CDS}.\mathsf{S}(\mathtt{x}, m_0, \mathtt{ct}^*_\mathsf{R}) \approx \mathtt{CDS}.\mathsf{S}(\mathtt{x}, m_1, \mathtt{ct}^*_\mathsf{R})$$

– ***Receiver simulation.*** *There exists a simulator* $\mathtt{CDS}.\mathsf{Sim}$*, such that for any polynomial-size distinguisher* D*, there exists a negligible* ϵ *such that for any security parameter* $\lambda \in \mathsf{N}$*, any* $\mathtt{x} \in \mathcal{L}$*, and* $\mathtt{w} \in \mathcal{R}_{\mathcal{L}}(\mathtt{x})$*,*

$$\mathtt{ct}_\mathsf{R} \approx \mathtt{CDS}.\mathsf{Sim}(\mathtt{x})$$

where $\mathtt{ct}_\mathsf{R} \leftarrow \mathtt{CDS}.\mathsf{R}(\mathtt{x}, \mathtt{w})$*.*

Notice that, CDS schemes can be instantiated assuming any two-message oblivious transfer protocol where the receiver message is computationally hidden from any semi-honest sender, and with (unbounded) simulation security against malicious receivers. Such oblivious transfer schemes are known based on DDH [NP01], Quadratic (or Nth) Residuosity [HK12], and LWE [BD18].

2.6 Universal Composability and Super-polynomial Simulation

We briefly review Universally composable (UC) security. For full details we refer to [Can01]. For the sake of completeness we include a short introduction of the notation of UC security in Appendix A.1.

Universal composability. UC security is a framework proposed by Canetti [Can01] as a way to define security for protocols such that security-preserving composition is possible. This allows for a modular design and analysis of protocols. For each cryptographic task, an ideal functionality can be defined, which incorporates the required properties of a protocol for the task and the allowed actions of an adversary. A protocol is said to securely realize the ideal functionality if, loosely speaking, any effect caused by an adversary attacking the protocol can be obtained by an adversary attacking the ideal functionality. When designing complex protocols, one can allow the involved parties to have secure access to ideal functionalities. Then, at the implementation phase, each ideal functionality is replaced by a protocol securely realizing the functionality. The composition theorem then guarantees security. For a complete overview, we refer to [Can01].

UC Security with super-polynomial simulation We next recall the relaxed notion of UC security by giving the simulator access to super-poly computational resources [GGJS12].

Definition 6. *A protocol* Π *UC-realizes an ideal functionality* $\mathcal{F}$ *in the hybrid model if, for every adversary* $\mathcal{A}$*, there exists a super-polynomial time simulator* Sim *such that for all environments* $\mathcal{Z}$*,* $\mathsf{IDEAL}^{\mathcal{F}}_{\mathsf{Sim}} \approx \mathsf{HYBRID}^{\mathcal{F}}_{\Pi,\mathcal{A}}$*. The protocol* Π *is statistically secure if the above definition holds for all unbounded* $\mathcal{Z}$

The universal composition theorem generalizes naturally to the case of UC-SPS, the details of which we refer [GGJS12].

Ideal functionality of two-party computation. As we mentioned before, the security of a protocol is analyzed by comparing what an adversary can do in the protocol to what it can do in an ideal scenario that is secure by definition. This is formalized by considering an ideal computation involving an incorruptible trusted third party to whom the parties send their inputs. The trusted party computes the functionality on the inputs and returns to each party its respective output. For the sake of completeness, we recall the ideal functionality of two-party protocols in Appendix A.1 and for more details refer to [LP07,GGJS12].

3 Concurrent Two-Party Computation in the Plain Model

3.1 High-Level Overview

In this section, we propose a generic construction of concurrent 2PC with two rounds in the plain model. Let $\mathtt{f}$ be any functionality. Consider two parties P_0 P_1 with inputs $\mathtt{x}_0$ ad $\mathtt{x}_1$ respectively who wish to compute $\mathtt{f}$ on their joint inputs by running a secure concurrent two-party computation protocol. Formally, we prove the following theorem:

Theorem 1. *Assuming the sub-exponential hardness of the LWE problem and the quantum sub-exponentially hard* $\mathtt{CCA}$*-secure commitments, then our two-rounds concurrently secure 2PC protocol with super-polynomial simulation for any functionality* $\mathtt{f}$ *that is secure against malicious adversaries in the plain model.*

Before describing our protocol formally, to help the exposition, we first give a brief overview of the construction and its security proofs in this subsection.

3.2 Construction

In this section we describe our concurrent two party construction. we first recall some notation and the primitives used in the construction.

Ingredients and notation:

- A (leveled) fully-homomorphic encryptions scheme $\mathtt{FHE}$ that contains the four algorithms ($\mathtt{FHE.Gen}$, $\mathtt{FHE.Enc}, \mathtt{FHE.Eval}, \mathtt{FHE.Dec}$) with maliciously circuit privacy. That is, even if both maliciously formed public key and ciphertext are used, encrypted outputs only reveal the evaluation of the circuit on some well-formed input $\mathtt{x}^*$.
- A $\mathtt{CCA}$-secure commitment scheme $\mathtt{CCA} = (\mathtt{CCA.Gen}, \mathtt{CCA.Com}, \mathtt{CCA.Open})$ that is a non-interactive $\mathtt{CCA}$ commitment scheme. It is secure against all adversaries running in time $\mathsf{T}^{\mathsf{Sec}}_{\mathtt{CCA.Com}}$, but can be broken by adversaries running in time $\mathsf{T}^{\mathsf{Brk}}_{\mathtt{CCA.Com}}$. Let $\mathtt{CCA.Ext}$ denote a brute force algorithm running in time $\mathsf{T}^{\mathsf{Brk}}_{\mathtt{CCA.Com}}$ that can break the commitment scheme. The $\mathsf{C}.\mathtt{CCA.Com}$ we use is tag-based. In the authenticated channels setting, the tag of each user performing a $\mathtt{CCA}$ commitment can just be its identity. In the general setting, each party can choose a strong digital signature verification key vk and signing key, and then sign all its messages using this signature scheme for every message sent in the protocol. This vk is then used as the tag for all $\mathtt{CCA}$ commitments. This ensures that every adversarial party must choose a tag that is different than any tags chosen by honest parties, otherwise the adversary will not be able to sign any of its messages by the existential unforgeability property of the signature scheme. This is precisely the property that is assumed when applying $\mathtt{CCA.Com}$. For ease of notation, we suppress writing the tags explicitly in our protocols below.
- A conditional disclosure of secrets scheme $\mathtt{CDS} = (\mathtt{CDS.R}, \mathtt{CDS.S}, \mathtt{CDS.D})$. That is, the sender holds an instance $\mathtt{x}_{\mathtt{CDS}}$ and message m and the receiver holds $\mathtt{x}_{\mathtt{CDS}}$ and a corresponding witness $\mathtt{w}_{\mathtt{CDS}}$. If the witness is valid, then the receiver obtains m, whereas if $\mathtt{x}_{\mathtt{CDS}} \notin \mathcal{L}_{\mathtt{CDS}}$, m remains hidden.
- $\mathsf{SPSS}.\mathtt{ZK} = (\mathtt{ZK.Gen}, \mathtt{ZK.P}, \mathtt{ZK.V})$ is a two message zero knowledge argument with super polynomial strong simulation (SPSS-ZK). The zero knowledge property holds against all adversaries running in time $\mathsf{T}_{\mathtt{ZK}}$. Let $\mathsf{Sim}_{\mathtt{ZK}}$ denote the simulator that produces simulated ZK proofs and let $\mathsf{T}_{\mathsf{Sim}}$ denote its running time.

Construction. In order to realize our protocol, we require that $\mathsf{poly}(\lambda) \ll \mathsf{T}_{\mathsf{Sim}} \ll \mathsf{T}^{\mathsf{Sec}}_{\mathtt{CCA.Com}} \ll \mathsf{T}^{\mathsf{Brk}}_{\mathtt{CCA.Com}} \ll \mathsf{T}_{\mathtt{ZK}}, \mathsf{T}_{\mathtt{FHE}}$. The construction of the protocol is described in Figure 1. In our construction, we use proofs for a some NP languages that we describe below.

NP language $\mathcal{L}_1$ is characterized by the following relation $\mathcal{R}_1$. And for statement $\mathtt{st} = (\mathtt{c}, \mathtt{cm}, \mathtt{ct})$ and witness: $\mathtt{w} = (\mathtt{x}, r)$, we say $\mathcal{R}_1(\mathtt{st}, \mathtt{w}) = 1$ if and only if:

$$\mathtt{c} = \mathtt{FHE.Enc}(\mathsf{pk}; \mathtt{x}; r) \ \wedge \mathtt{cm} = \mathtt{CCA.Com}(\mathtt{x}; r)$$

where $\mathtt{c}$ and $\mathtt{cm}$ form a $\mathtt{FHE}$ encryption of $\mathtt{x}_i$ and a non-malleable commitment of $(\mathtt{x}; r)$ where $\mathtt{x}$ is a message and r is randomness. NP language $\mathcal{L}_2$, is characterized by the following relation $\mathcal{R}_2$. And for

statement $\mathtt{st} = (\mathtt{c}', \mathtt{cm}, \mathtt{ct})$ and witness: $\mathtt{w} = (\mathtt{x}; r)$, where $\mathtt{c}' = \mathtt{FHE.Enc}(\mathtt{pk}', \mathtt{x}'; r')$ and $\mathtt{ct_R} = \mathtt{CDS.S}(\mathtt{c}', \mathtt{w}')$ we say $\mathcal{R}_2(\mathtt{st}, \mathtt{w}) = 1$ if and only if:

$$\mathtt{c} = \mathtt{FHE.Eval}(\mathtt{pk}', \mathtt{c}', \mathtt{f}(\mathtt{x}, .); r) \ \wedge \mathtt{cm} = \mathtt{CCA.Com}(\mathtt{x}; r) \ \wedge \mathtt{ct} = \mathtt{CDS.S}(\mathtt{c}', \mathtt{c}, \mathtt{ct_R})$$

where $\mathtt{c}$ and $\mathtt{cm}$ form a FHE encryption of $\mathtt{x}$ and a non-malleable commitment of $(\mathtt{x}; r)$ where $\mathtt{x}$ is a message and r is randomness and $\mathtt{f}$ is a function that takes two inputs of size $|\mathtt{x}|$ and the input $\mathtt{x}$ is fixed into the function and $\mathtt{c}'$ is a FHE encryption of some message $\mathtt{x}'$ under the public-key $\mathtt{pk}'$ and we want $\mathtt{c}$ to be the FHE encryption of $\mathtt{f}(\mathtt{x}, \mathtt{x}')$ under $\mathtt{pk}'$.

We also define another NP language $\mathcal{L}_3$ with the corresponding relation $\mathcal{R}_3$, which we later use a zero knowledge proof on this language to show that same message has been used an FHE encryptions and also hard-coded in the functions in its FHE evaluation . namely assume $\mathtt{c}$ is the encryption of $\mathtt{x}$, and $\mathtt{c}'$ is some other FHE encryption under public key $\mathtt{pk}'$. And $\mathtt{f}(\cdot, \cdot)$ is a function that takes two inputs if we fix input $\mathtt{x}$ into it, it becomes a single input function $\mathtt{f}(\mathtt{x}, \cdot)$. We define statements of $\mathcal{L}_3$ as $\mathtt{st} = (\mathtt{c}_1, \mathtt{c}_2)$ and for witness $\mathtt{w} = (\mathtt{x}, r, r')$, where $\mathtt{c}_1 = \mathtt{FHE.Enc}(\mathtt{pk}, \mathtt{x}_1, r)$ and $\mathtt{c}_2 = \mathtt{FHE.Eval}(\mathtt{pk}', \mathtt{c}', \mathtt{f}(\mathtt{x}_2, \cdot); r')$ We say $\mathcal{R}_3(\mathtt{c}_1, \mathtt{c}_2) = 1$ if and only if $\mathtt{x}_1 = \mathtt{x}_2$.

Finally we define NP language $\mathcal{L}$ as the the AND of $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$.

The correctness of the protocol follows from the correctness of the CCA-secure commitment scheme CCA, the conditional disclosure of secrets CDS, and the zero knowledge proof system SPSS.ZK

High level construction. Now we describe the our construction at a high level. Parties P_0 and P_1 with corresponding inputs $\mathtt{x}_0$ and $\mathtt{x}_1$ want to compute the output of the function $\mathtt{f}(\mathtt{x}_0, \mathtt{x}_1)$ in a secure way. Our protocol has two rounds, were in each round, each party sends a message to the other party in a simultaneous manner. For the first round of the protocol, each party, first generates their randomness and runs the setup algorithm for an FHE scheme which they will be using throughout the protocol. Then they compute the FHE encryption of their inputs, let us call it $\mathtt{c}_i$, they also commit to their inputs using a CCA-secure commitment scheme, and compute committed values $\mathtt{cm}_i$ for $i \in \{0, 1\}$. Then each party, runs the receiver's algorithm of the conditional disclosure of secrets scheme (CDS.R) on the encrypted ciphertext and some corresponding witness $\mathtt{w}$ with respect to the NP language $\mathcal{L}$ that we defined above. Then each party plays the role of the verifier in a SPSS zero knowledge schemes and gets the verifier's parameter ver and a state zkst. At the end of the first round, each party sends their encryption ciphertext, commitment, verifier's parameter and the CDS ciphertext as their message to the other party.

In the second round, parties run virtually same algorithms according to their inputs simultaneously. first, they run the FHE evaluation of the other other partiy's ciphertext and public key, on the function $\mathtt{f}(\mathtt{x}_i, \cdot)$ which is the ultimate function they want to compute in the protocol, in a way that the party's input is fixed in the function. Finally they get the ciphertext $\hat{\mathtt{c}}_i$ with is the encryption of $\mathtt{f}(\mathtt{x}_0, \mathtt{x}_1)$ under the public key of the other party. Then they commit to their input, using the randomness generated in the second round, and run the sender's algorithm $\mathtt{CDS.S}(\mathtt{c}_{1-i}, \hat{\mathtt{c}}_i, \mathtt{ct}_{1-i})$ and gets the corresponding CDS sender output $\hat{\mathtt{ct}}_i$.

Then each party provides a zero-knowledge proof, where it proves the consistency of the commitment and the CDS cipher text and the evaluated FHE ciphertext; they also prove that they used same input on both FHE encryptions. That is, a zero-knowledge proof for *AND* of $\mathcal{L}_2, \mathcal{L}_3$. Then they send the commitment and CDS ciphertext as well as the zero-knowledge proof to the other party.

Finally, each party after receiving the messages from the second round, each party locally checks the zero-knowledge proof and if it was correct, they use the CDS ciphertext that they received in the second round $\hat{\mathtt{ct}}_{1-i}$ and their own CDS key k_i and run the CDS decryption algorithm $\mathtt{CDS.D}(k_i, \hat{\mathtt{ct}}_{1-i})$ to get an FHE ciphertext which they can later decrypt it using their own FHE secret-key to get the output of $\mathtt{f}(\mathtt{x}_0, \mathtt{x}_1)$

High level security proof. Here we describe a high-level overview of the security proof. Let's consider an adversary $\mathcal{A}$ who corrupts the parties. Recall that the goal is to move from the real world to the ideal world such that the outputs of the honest parties along with the view of the adversary is indistinguishable. We reach this with a sequence of computationally indistinguishable games. The first game Game_1, refers to the real world. In Game_2, the simulator extracts the adversary's input and randomness (used in the protocol) by a brute force break of the non-malleable commitment. The simulator aborts if the extracted values don't reconstruct the protocol messages for the underlying malicious protocol correctly. These two games are

Inputs party P_0 and P_1 have $\mathtt{x}_0$ and $\mathtt{x}_1$ as their inputs

- **Round 1:**
  - $i \in \{0,1\}$
    Each party P_i does the following:
  - $r_i \leftarrow\$ \mathbb{Z}_p$
  - $\mathtt{c}_i \leftarrow \mathtt{FHE.Enc}(\mathsf{pk}_i, \mathtt{x}_i; r_i)$
  - $\mathtt{cm}_i \leftarrow \mathtt{CCA.Com}(\mathtt{x}_i, r_i)$
  - $(\mathtt{ct}_i, k_i) \leftarrow \mathtt{CDS.R}(\mathtt{c}_i, \mathtt{w}_i)$
  - $(\mathsf{ver}_i, \mathsf{zkst}_i) \leftarrow \mathtt{ZK}(1^\lambda)$
    P_i sends $(\mathtt{c}_i, \mathtt{cm}_i, \mathtt{ct}_i, \mathsf{ver}_i, \mathsf{pk}_i)$
- **Round 2:**
    Each party P_i does the following:
  - $\hat{r}_i \leftarrow\$ \mathbb{Z}_p$
  - $\hat{\mathtt{c}}_i \leftarrow \mathtt{FHE.Eval}(\mathsf{pk}_{1-i}, \mathtt{c}_{1-i}, \mathtt{f}(\mathtt{x}_i, \cdot); \hat{r}_i)$
  - $\hat{\mathtt{cm}}_i \leftarrow \mathtt{CCA.Com}(\mathtt{x}_i, \hat{r}_i)$
  - $\hat{\mathtt{ct}}_i \leftarrow \mathtt{CDS.S}(\mathtt{c}_{1-i}, \hat{\mathtt{c}}_i, \mathtt{ct}_{1-i})$
  - $\pi_{\mathtt{zk},i} \leftarrow \mathtt{ZK.P}(\mathsf{ver}_i, \mathtt{c}_i, \hat{\mathtt{c}}_i, \hat{\mathtt{cm}}_i, \hat{\mathtt{ct}}_i, \mathtt{w}_i)$
  - P_i sends $\hat{\mathtt{cm}}_i, \hat{\mathtt{ct}}_i, \pi_{\mathtt{zk},i}$
- **Computation part**
  Each party does the following:
- if $\mathtt{ZK.V}(\mathsf{zkst}_i, \pi_{\mathtt{zk},1-i}) = 1$ returns $\mathtt{FHE.Dec}(\mathsf{sk}_i, \mathtt{CDS.D}(k_i, \hat{\mathtt{ct}}_{1-i}))$.

Fig. 1. Generic construction of concurrent two-party computation.

indistinguishable because from the soundness of the proof system, except with negligible probability, the values extracted by the simulator correctly reconstruct to protocol messages.

Then, in Game_3, we switch the SPSS.ZK arguments used by an honest party in rounds 2 to simulated ones. This game is computationally indistinguishable from the previous game by the security of the SPSS.ZK system. Notice that when we switch from real to simulated arguments, we can no longer rely on the adversary's zero knowledge arguments to argue the correctness of the values extracted by breaking the CCA.Com scheme. That is, the adversary's arguments may not be simulation sound. However, recall that to check the validity of the extracted values, we only rely on the correct reconstruction of the semi-malicious protocol messages, and hence this is not a problem. Also, the running time of the simulator in these two hybrids is the time taken to break the non-malleable commitment $\mathsf{T}^{\mathsf{Brk}}_{\mathtt{CCA.Com}}$ (which must be lesser than the time against which the zero knowledge property holds, $\mathsf{T}_{\mathtt{ZK}}$.).

In Game_4, we switch the non-malleable commitments sent by honest party to be commitments of 0 instead of the actual input and randomness. Recall that since the arguments of the honest parties are simulated, this doesn't violate correctness. Also, this game is computationally indistinguishable from the previous game by the security of the non-malleable commitment scheme. One issue that arises here is whether the simulator continues to extract the adversary's inputs correctly. Recall that to extract, the simulator has to break the non-malleable commitment for which it has to run in time $\mathsf{T}^{\mathsf{Brk}}_{\mathtt{CCA.Com}}$. However, then the reduction to the security of the non-malleable commitment only makes sense if the simulator runs in time lesser than that needed to break the non-malleable commitment. We overcome this issue by a sequence of sub-games where we first switch the simulator to not extract the adversary's inputs, then switch the non-malleable commitments and then finally go back to the simulator extracting the adversary's inputs. We elaborate on this in the formal proof.

Then, in Game_5 we run the simulator of the protocol using the extracted values to generate the protocol messages. This game is indistinguishable from the previous one by the security of the concurrent 2PC. Once again, in order to ensure correctness of the extracted values, we require the running time of the simulator - which is $\mathsf{T}^{\mathsf{Brk}}_{\mathtt{CCA.Com}}$ to be lesser than the time against which the malicious protocol concurrent 2PC is secure. This is because, then, the simulator can continue to extract the adversary's message and randomness used

for the protocol by breaking the malicious protocol. Now, Game_5 corresponds to the ideal world. Notice that our simulation is in fact straight-line. There are other minor technicalities that arise and we elaborate on this in the formal proof.

3.3 Security Proof

In this section, we formally prove Theorem 1. Consider an adversary $\mathcal{A}$ who corrupts a party in the protocol. For each party P_i, let's say that the size of input and randomness used in the protocol is $p(\lambda)$ for some polynomial p. That is, $|(\mathtt{x}_i, r_i)| = p(\lambda)$. The strategy of the simulator Sim against a malicious adversary $\mathcal{A}$ is described in Figure 2.

- **Round 1:** for each party P_i for $i \in \{0,1\}$, Sim does as follows:
  - Compute $\mathtt{c}_i \leftarrow \mathtt{FHE.Enc}(\mathsf{pk}_i, \vec{0}, r_i)$ for $r_i \leftarrow\$ \{0,1\}^\lambda$ and $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathtt{FHE.Gen}(1^\lambda, 1^{s(\lambda)})$.
  - Compute $\mathtt{cm}_i \leftarrow \mathtt{CCA.Com}(\vec{0}||r_i)$ and $(\mathtt{ct}_i, k) \leftarrow \mathtt{CDS.R}(\mathtt{c}_i, \mathtt{w}_i)$ where as before $\mathtt{w}_\mathsf{R}$ is such that $\mathtt{c}_i$ and $\mathtt{cm}_i$ are well-formed.
  - Initiates the execution of the SPSS.ZK argument system as: $(\mathsf{ver}_i, \mathsf{zkst}_i) \leftarrow \mathtt{ZK}(\lambda)$.
  - For each honest party P_i do:
- **Input Extraction:** Sim does the following:
  For each honest party P_i do:
  - Break $\mathtt{CCA.Com}_{1-i}$ and obtains $(\mathtt{x}_{1-i}, r_{1-i}) \leftarrow \mathtt{CCA.Ext}(\mathtt{cm}_{1-i})$. That is the input and the randomness of party P_{1-i} seen by party P_i.
  - This step takes time $\mathsf{T}^{\mathsf{Brk}}_{\mathtt{CCA.Com}}$
  - Initialize a variable $\mathsf{correct} = 1$. Then, for each malicious party P_{1-i} , do
  - Set $\mathsf{correct} = 0$ if the set of values $(\mathtt{x}_{1-i}, r_{1-i})$, for the i corresponding to honest parties P_i is not equal. Let R denote the set of $(\mathtt{x}_{1-i}, r_{1-i})$
  - Set $\mathsf{correct} = 0$ if $\mathtt{c}_{1-i} \neq \mathtt{FHE.Enc}(\mathsf{pk}_{1-i}, \mathtt{x}_{1-i}, r_{1-i})$.
- **Round 2:** Let τ^1 denote the protocol transcript after round 1. Sim does the following:
  - For each honest party P_i:
  - If $\mathsf{correct} = 1$, compute $\hat{\mathtt{c}}_i \leftarrow \mathtt{FHE.Sim}(\mathsf{pk}_{1-i}, \mathtt{c}_{1-i}, \vec{y}, \hat{r}_i, \tau^1)$ where $\hat{r}_i \leftarrow\$ \{0,1\}^*$ and $\vec{y}$ is given by the ideal functionallity.
  - such that $\vec{y} \leftarrow F(\mathtt{x}_i, \mathtt{x}_{1-i})$ for some function F.
  - Else, compute $\hat{\mathtt{c}}_i$ as before but by setting $\vec{y} = \vec{0}$.
  - And compute $\hat{\mathtt{cm}}_i \leftarrow \mathtt{CCA.Com}(\vec{0}||\hat{r}_i)$ and $\hat{\mathtt{ct}}_i \leftarrow \mathtt{CDS.S}(\mathtt{c}_i, \hat{\mathtt{st}}_i)$.
  - Compute $\hat{\pi}_{\mathsf{zk},i} \leftarrow \mathsf{Sim}_{\mathsf{zk}}(\mathsf{ver}_{1-i}, \mathsf{st}_i)$ and $\hat{\pi}'_{\mathsf{zk},i} \leftarrow \mathsf{Sim}_{\mathsf{zk}}(\mathsf{ver}_{1-i}, \mathsf{st}'_i)$ respectively for $\hat{\mathsf{st}}_i = (\hat{\mathtt{c}}_i, \hat{\mathtt{cm}}_i, \hat{\mathtt{ct}}_i) \in \hat{\mathcal{L}}_i$ and $\hat{\mathsf{st}}'_i = (\mathtt{c}_i, \hat{\mathtt{c}}_i) \in \hat{\mathcal{L}}'_i$.
  - Observe that this takes time $\mathsf{T}_{\mathsf{Sim}_{\mathsf{zk}}}$.
  - Send $(\hat{\mathtt{c}}_i, \hat{\mathtt{cm}}_i, \hat{\pi}_{\mathsf{zk},i})$ to P_{1-i}.
- **Abort Phase:**
  - For each malicious party P_{1-i}
  - Output "Abort" if $\mathsf{correct} = 0$.
  - Also, output "Abort" if $\hat{\mathtt{c}}_{1-i} \neq \mathtt{FHE.Eval}(\mathsf{pk}_i, \mathtt{c}_i, \mathtt{f}, \hat{r}_{1-i})$.
- **Output Computation:**
  - Sim does the following:
  - For each honest party P_i abort if $\mathtt{ZK.V}(\mathsf{zkst}_i, \hat{\pi}_{\mathsf{zk},1-i}) \neq 1$.
  - Else, instruct the ideal functionality to deliver output to the honest parties.

Fig. 2. Simulation strategy in the concurrent 2PC

We now show that the simulation strategy described in Figure 2 is successful against all malicious PPT 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. We will show this via a series of computationally indistinguishable games where the first game Game_1 corresponds to the real world and the last game corresponds to the ideal world.

1. Game_1: In this game, consider a simulator $\mathsf{Sim}_{\mathsf{Game}}$ that plays the role of the honest parties. $\mathsf{Sim}_{\mathsf{Game}}$ runs in polynomial time.

2. Game_2: In this game, the simulator $\mathsf{Sim}_{\mathsf{Game}}$ also runs the "Input Extraction" phase and the "Abort" phase as in Figure 2. $\mathsf{Sim}_{\mathsf{Game}}$ runs in time $\mathsf{T}^{\mathsf{Brk}}_{\mathtt{CCA.Com}}$.

3. Game_3: This game is identical to the previous game except that in Round 2, $\mathsf{Sim}_{\mathsf{Game}}$ now computes simulated SPSS.ZK proofs as done in Figure 2. $\mathsf{Sim}_{\mathsf{Game}}$ runs in time $\mathsf{T}^{\mathsf{Brk}}_{\mathtt{CCA.Com}}$.

4. Game_4: This is identical to the previous game except that $\mathsf{Sim}_{\mathsf{Game}}$ now computes all $\mathtt{cm}_i$ and $\hat{\mathtt{cm}}_i$ as $\mathtt{CCA}$ commitment of 0. $\mathsf{Sim}_{\mathsf{Game}}$ runs in time $\mathsf{T}^{\mathsf{Brk}}_{\mathtt{CCA.Com}}$.

5. Game_5: This game is identical to the previous game except that in Round 2, $\mathsf{Sim}_{\mathsf{Game}}$ now computes the messages of the protocol using the simulator algorithms $\mathtt{FHE.Sim}$ as done by Sim in the ideal world. Sim also instructs the ideal functionality to deliver outputs to the honest parties as done by Sim. This game is now same as the ideal world.

We now show that every pair of successive games is computationally indistinguishable.

Lemma 1. *Assuming soundness of the SPSS.ZK argument system, binding of the non-interactive CCA-commitment scheme and correctness of the* $\mathtt{FHE}$ *scheme,* Game_1 *is computationally indistinguishable from* Game_2.

Proof. The only difference between the two games is that in Game_2, $\mathsf{Sim}_{\mathsf{Game}}$ may output "Abort" which doesn't happen in Game_1. More specifically, in Game_2, "Abort" occurs if event E described below is true.

Event E: Is true if : For any malicious party P_{1-i}, (i) the SPSS.ZK proof sent by P_{1-i} in round 2 verify correctly, and (ii) either of the following occur:

(ii-1) The set of values $(\mathtt{x}_{1-i}, r_{1-i})$ that are committed to using the non-malleable commitment $\mathtt{CCA.Com}$ are not same for honest party P_i.

(ii-2) OR $\mathsf{c}_{1-i} \neq \mathtt{FHE.Enc}(\mathsf{pk}_{1-i}, \mathtt{x}_{1-i}; r_{1-i})$;

(ii-3) OR $\hat{\mathsf{ct}}_{1-i} \neq \mathtt{CDS.S}(\hat{\mathsf{c}}_{1-i}, \hat{\mathsf{st}})$.

That is, in simpler terms, the event E occurs if for any malicious party, it gives valid ZK proofs in round 2 but its protocol statement is not consistent with the values it committed to. Therefore, in order to prove the indistinguishability of the two games, it is enough to prove the lemma below.

Sub-Lemma1. Pr[Event E occurs in Game_2] = $\mathsf{negl}(\lambda)$.

Proof. Suppose the event E does occur. From the binding property of the commitment scheme and the correctness of the $\mathtt{FHE}$ scheme in Figure 1, observe that if any of the above conditions are true, it means there exists some statements $\mathsf{st}_{2,1-i} \notin \mathcal{L}_{2,1-i}$ or $\mathsf{st}_{3,1-i} \notin \mathcal{L}_{3,1-i}$ where P_i is honest and P_{1-i} is malicious. However, the proof for the statement verified correctly which means that the adversary has produced a valid proof for a false statement. This violates the soundness property of the SPSS.ZK argument system which is a contradiction. □

□

Lemma 2. *Assuming the zero knowledge property of the SPSS.ZK argument system,* Game_2 *is computationally indistinguishable from* Game_3.

Proof. The only difference between the two games is that in Game_2, $\mathsf{Sim}_{\mathsf{Game}}$ computes the proofs in Round 2 honestly, by running the algorithm $\mathtt{ZK}$ of the SPSS.ZK argument system, whereas in Game_3, a simulated

proof is used. If the adversary $\mathcal{A}$ can distinguish between the two games, we can use $\mathcal{A}$ to design an algorithm $\mathcal{A}_{\mathsf{zk}}$ that breaks the zero knowledge property of the argument system.

Suppose the adversary can distinguish between the two games with non-negligible probability $p_{\mathcal{A}}$. Then, by a simple hybrid argument, there exists games $\mathsf{Game}_{2,k}$ and $\mathsf{Game}_{2,k+1}$ that the adversary can distinguish with non-negligible probability $p'_{\mathcal{A}} < p_{\mathcal{A}}$ such that: the only difference between the two games is in the proof sent by an honest party P_i to a (malicious) party P_{1-i} in one of the round 2.

$\mathcal{A}_{\mathsf{zk}}$ performs the role of $\mathsf{Sim}_{\mathsf{Game}}$ in its interaction with $\mathcal{A}$ and performs all the steps exactly as in $\mathsf{Game}_{2,k}$ except the proof in Round 2 sent by P_i to P_{1-i} . It interacts with a challenger C of the SPSS.ZK argument system and sends the first round message ver it received from the adversary. $\mathcal{A}_{\mathsf{zk}}$ receives from C a proof that is either honestly computed or simulated. $\mathcal{A}_{\mathsf{zk}}$ sets this received proof as its message $\pi_{\mathsf{zk},i}$ in Round 2 of its interaction with $\mathcal{A}$. In the first case, this exactly corresponds to $\mathsf{Game}_{2,k}$ while the latter exactly corresponds to $\mathsf{Game}_{2,k+1}$. Therefore, if $\mathcal{A}$ can distinguish between the two games, $\mathcal{A}_{\mathsf{zk}}$ can use the same distinguishing guess to distinguish the proofs: i.e, decide whether the proofs received from C were honest or simulated. Now, notice that $\mathcal{A}_{\mathsf{zk}}$ runs only in time $\mathsf{T}^{\mathsf{Brk}}_{\mathtt{CCA.Com}}$ (during the input extraction phase), while the SPSS.ZK system is secure against adversaries running in time $\mathsf{T}_{\mathtt{ZK}}$. Since $\mathsf{T}^{\mathsf{Brk}}_{\mathtt{CCA.Com}} < \mathsf{T}_{\mathtt{ZK}}$, this is a contradiction and hence proves the lemma. In particular, this also means the following: Pr[Event E occurs in Game_3] = $\mathsf{negl}(\lambda)$. □

Lemma 3. *Let* CCA.Com *is a* CCA*-secure commitment scheme, then* Game_3 *is computationally indistinguishable from* Game_4.

Proof. We prove this using a series of computationally indistinguishable intermediate games as follows.

$\mathsf{Game}_{3,1}$: This is same as Game_3 except that the simulator $\mathsf{Sim}_{\mathsf{Game}}$ does not run the input extraction phase apart from verifying the SPSS.ZK proofs. Also, $\mathsf{Sim}_{\mathsf{Game}}$ does not run the "Abort" phase (in particular, no breaking or extraction algorithm is run and there is no "Abort"). In this game, $\mathsf{Sim}_{\mathsf{Game}}$ runs in time $\mathsf{T}_{\mathsf{Sim}_{\mathsf{zk}}}$ which is lesser than $\mathsf{T}^{\mathsf{Brk}}_{\mathtt{CCA.Com}}$.

$\mathsf{Game}_{3,2}$: This is identical to the previous game except that, $\mathsf{Sim}_{\mathsf{Game}}$ now computes all the messages $\mathtt{cm}_{\mathsf{R}}$ (and $\mathtt{cm}_{\mathsf{S}}$) as non-malleable commitments of 0 as it is done in the ideal world. In this game also $\mathsf{Sim}_{\mathsf{Game}}$ runs in time $\mathsf{T}_{\mathsf{Sim}_{\mathsf{zk}}}$.

$\mathsf{Game}_{3,3}$: This is same as Game_3 except that the simulator does run the input extraction phase and Abort phase. It is easy to see that $\mathsf{Game}_{3,3}$ is the same as Game_4. Here, $\mathsf{Sim}_{\mathsf{Game}}$ runs in time $\mathsf{T}^{\mathsf{Brk}}_{\mathtt{CCA.Com}}$ which is greater than $\mathsf{T}_{\mathsf{Sim}_{\mathsf{zk}}}$.

Now, We prove the indistinguishability of these intermediate games and this completes the proof of the lemma 3.

1-Sub-Lemma 3. The game Game_3 is statistically indistinguishable from $\mathsf{Game}_{3,1}$.

Proof. The only difference between the two games is that in Game_3, the simulator might output Abort which doesn't happen in $\mathsf{Game}_{3,1}$. As shown in the proof of Lemma 2, the probability that Event E occurs in Game_3 is negligible. This means that the probability that the simulator outputs "Abort" in Game_3 is negligible and this completes the proof. □

2-Sub-Lemma 3. Assuming the CCA property of the CCA commitment scheme CCA.Com, $\mathsf{Game}_{3,1}$ is computationally indistinguishable from $\mathsf{Game}_{3,2}$.

Proof. The only difference between the two games is that in $\mathsf{Game}_{3,1}$, for every honest party P_i, $\mathsf{Sim}_{\mathsf{Game}}$ computes the commitment messages $\mathtt{cm}_i$ as a commitment of $\mathtt{x}_i$ and r_i, whereas in $\mathsf{Game}_{3,2}$, they are computed as a commitment of 0. If the adversary $\mathcal{A}$ can distinguish between the two games, we can use $\mathcal{A}$ to design an algorithm $\mathcal{A}_{\mathtt{CCA.Com}}$ that breaks the security of the CCA property of the CCA commitment scheme CCA.Com.

In a way that $\mathcal{A}_{\mathtt{CCA.Com}}$ acts as the man-in-the-middle adversary interacting with a challenger C. $\mathcal{A}_{\mathtt{CCA.Com}}$ also plays the role of $\mathsf{Sim}_{\mathsf{Game}}$ in its interaction with the adversary $\mathcal{A}$ (also due to the CCA security definition, $\mathcal{A}_{\mathtt{CCA.Com}}$ also has access to the opening oracle of the commitment). It generates all the messages except the messages $\mathtt{cm}_i$ exactly as done by $\mathsf{Sim}_{\mathsf{Game}}$ in $\mathsf{Game}_{3,1}$. Corresponding to the CCA definition, $\mathcal{A}_{\mathtt{CCA.Com}}$ has to send the messages (m_0, m_1) to the oracle C, which here she sets $m_0 = 0$. Then after receiving the challenge from C, $\mathcal{A}_{\mathtt{CCA.Com}}$ uses this challenge to feed the adversary $\mathcal{A}$. Then she returns the output of $\mathcal{A}$.

Now, we can see that in the first case, when C generates commitments to $\mathtt{x}_i$ and r_i, $\mathcal{A}$'s view corresponds to $\mathsf{Game}_{3,1}$ while in the latter case, it exactly corresponds to $\mathsf{Game}_{3,2}$. However, from the security of the $\mathtt{CCA.Com}$ scheme, the joint distribution of the value committed to by the adversary $\mathcal{A}_{\mathtt{CCA.Com}}$ (which is the same as $\mathcal{A}$'s commitments) and its view must be indistinguishable in both cases. Therefore, if $\mathcal{A}$ can distinguish between the two games, then $\mathcal{A}_{\mathtt{CCA.Com}}$ can break the $\mathtt{CCA}$ property of the $\mathtt{CCA}$ commitment scheme $\mathtt{CCA.Com}$. However, $\mathcal{A}_{\mathtt{CCA.Com}}$ only runs in time $\mathsf{T}_{\mathsf{Sim}_{\mathsf{zk}}} < \mathsf{T}^{\mathsf{sec}}_{\mathtt{CCA.Com}}$ and hence this is a contradiction, thus proving the sub-lemma. In addition, notice that since the joint distribution of the adversary $\mathcal{A}$'s committed values and his view is indistinguishable in both games, this implies that Event E still occurs only with negligible probability in $\mathsf{Game}_{3,2}$ as well. □

3-Sub-Lemma 3. $\mathsf{Game}_{3,2}$ is statistically indistinguishable from $\mathsf{Game}_{3,3}$.

Proof.. The only difference between the two games is that in $\mathsf{Game}_{3,3}$, the simulator might output Special Abort which doesn't happen in $\mathsf{Game}_{3,2}$. As shown in the proof of *2-Sub-Lemma 3*, the probability that Event E occurs in $\mathsf{Game}_{3,2}$ is negligible. This means that the probability that the simulator outputs Special Abort in $\mathsf{Game}_{3,3}$ is negligible and this completes the proof. □

□

Lemma 4. *Assuming the security of* $\mathtt{FHE}$ *scheme in Figure 1,* Game_4 *is computationally indistinguishable from* Game_5.

Proof. The only difference between the two games is that in Game_4, $\mathsf{Sim}_{\mathsf{Game}}$ computes the messages of protocol in Figure 1 correctly using the honest parties inputs, whereas in Game_5, in each session, if $\mathsf{correct} = 1$, they are computed by running the simulator $\mathtt{FHE.Sim}$ for $\mathtt{FHE}$ and if $\mathsf{correct} = 0$, they are computed using the honest parties' strategy. Therefore, the only difference in any session is if $\mathsf{correct} = 1$. If the adversary $\mathcal{A}$ can distinguish between the two hybrids, we will use $\mathcal{A}$ to design an algorithm $\mathcal{A}_{\mathtt{FHE}}$ that can break the security of $\mathtt{FHE}$. Suppose $\mathcal{A}$ can distinguish between these two hybrids with some non-negligible probability p. Then by a simple hybrid argument, there exists games $\mathsf{Game}_{4,s}$ and $\mathsf{Game}_{4,s+1}$ that can be distinguished by $\mathcal{A}$ with some non-negligible probability $p' < p$ where the difference between these two games is that only in session s, the protocol messages are computed differently. $\mathcal{A}_{\mathtt{FHE}}$ interacts with a challenger C to break the security of $\mathtt{FHE}$. Also, $\mathcal{A}_{\mathtt{FHE}}$ performs the role of $\mathsf{Sim}_{\mathsf{Game}}$ in its interaction with the adversary A exactly as in $\mathsf{Game}_{4,s}$ except for session s. Whatever parties $\mathcal{A}$ wishes to corrupt in session s, $\mathcal{A}_{\mathtt{FHE}}$ corrupts the same parties in its interaction with $\mathtt{FHE}$. Similarly, whatever messages $\mathcal{A}$ sends to $\mathcal{A}_{\mathtt{FHE}}$ as part of $\mathtt{FHE}$ in session s that correspond to $\mathtt{FHE}$ messages, $\mathcal{A}_{\mathtt{FHE}}$ sends the same messages to the challenger C. Now, whatever messages C sends, $\mathcal{A}_{\mathtt{FHE}}$ forwards the same to the adversary $\mathcal{A}$ as its messages for the $\mathtt{FHE}$ in session s.

Observe that $\mathcal{A}_{\mathtt{FHE}}$ runs in time $\mathsf{T}^{\mathsf{Brk}}_{\mathtt{CCA.Com}}$. If C sends messages that are computed correctly, this exactly corresponds to $\mathsf{Game}_{4,s}$ in $\mathcal{A}_{\mathtt{FHE}}$'s interaction with $\mathcal{A}$. On the other hand, if C sends simulated messages, this exactly corresponds to $\mathsf{Game}_{4,s+1}$. Therefore, if $\mathcal{A}$ can distinguish between these two games, $\mathcal{A}_{\mathtt{FHE}}$ can use the same distinguishing guess to break the security $\mathtt{FHE}$ in Figure 1. However, this protocol is secure against all adversaries running in time $\mathsf{T}_{\mathtt{FHE}}$, where $\mathsf{T}_{\mathtt{FHE}} > \mathsf{T}^{\mathsf{Brk}}_{\mathtt{CCA.Com}}$ and hence this is a contradiction. This completes the proof of the lemma.

□

4 Applications of Concurrent Two-Party Computation

In this section we discuss applications of our concurrent two-party computation (2PC) scheme to Password-Authenticated Key-Exchange (PAKE) in the plain model as well as concurrent blind signature.

4.1 Concurrent PAKE in the Pain Model

We show how the concurrent two-party computation (2PC) in Figure 1 enables us to construct Password-Authenticated Key-Exchange (PAKE) in the plain model. Briefly in a PAKE protocol, each user U owns a word $\mathtt{x}_1$ in a certain language $\mathcal{L}_1$ and expects the other user to own a word $\mathtt{x}_2$ in a language $\mathcal{L}_2$. If everything

is compatible (i.e., the languages are the expected languages and the words are indeed in the respective languages), the users compute a common high-entropy secret key, otherwise they learn nothing about the other user's values.

We first observe that our concurrent 2PC in Figure 1 directly gives us a concurrently secure PAKE construction in the plain model. More precisely, let each party P_i in the concurrent 2PC in Figure 1 plays the role as the user U (described above) in the PAKE construction in way that by having an input $\mathtt{x}_i$, each party can compute $\mathtt{f} \leftarrow \mathtt{f}(\mathtt{x}_0, \mathtt{x}_1)$ as the a common high-entropy secret key. We note that before this work, even two-round PAKE schemes (not concurrent) in the plain model were left an open problem.

Security of PAKEs: In the cryptography literature there are two leading paradigms for rigorously defining the above intuition. The first is the so-called "game-based" definition introduced by Bellare et al. [BPR00]. In this definition, a password is chosen from a distribution with min-entropy k, and the security experiment considers an interaction 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 advantage better than $Q \cdot 2^{\lambda}$ plus a negligible quantity where Q the number of online attacks by the adversary i.e., actively interferes in Q sessions of the protocol and can make at most Q password guesses.

A second approach uses a "simulation-based" definition [BMP00,CHK$^+$05]. The most popular choice here is to work in the framework of universal composability (UC) [Can01], and this is what we mean from now on when we refer to a simulation-based definition. This approach works by first defining an appropriate ideal functionality for PAKE; a PAKE protocol is then considered secure if it realizes that functionality. Canetti *et al.* [CHK$^+$05] pursued this approach, and defined a PAKE functionality that explicitly allows an adversary to make password guesses; a random session key is generated unless the adversary's password guess is correct. As argued by Canetti *et al.* [CHK$^+$05], this approach has a number of advantages. A UC definition ensures security under arbitrary protocol composition, which is useful for arguing security of protocols that use PAKE as a subroutine, e.g., for converting symmetric PAKE to asymmetric PAKE [GMR06,HJK$^+$18] or strong asymmetric PAKE [JKX18]. This is especially important in the context of PAKE standardization, because strong asymmetric PAKE protocols can strengthen the current practice of password-over-TLS authentication while achieving optimal security in the face of server compromise.

In context of UC security, following the security definitions of PAKE [BMP00], in general PAKE protocols must satisfy: (i) *completeness* meaning that for any real world adversary that faithfully passes messages between two user instances with complimentary roles and identities, both user instances accept, and (ii) *simulatability* meaning that for every efficient real world adversary $\mathcal{A}$, there exists an simulator Sim such that real-word and ideal-world are computationally indistinguishable, where real-world and ideal-world denote the transcript of $\mathcal{A}$'s and Sim's operations respectively. We note that the ideal functionality of PAKEs is similar to the ideal functionality of concurrent two-party computation (see Section 2.6).

Security of our concurrent PAKE. Now, in Theorem 2 we analyze security of our concurrent PAKE in the UC-secure model with super-polynomial simulation as explained in 2.6. Recalling that this is UC security by giving the simulator access to super-poly computational resources [GGJS12].

Theorem 2. *Let the concurrent 2PC in Figure 1 be secure against malicious adversaries in the plain model. Then the PAKE protocol* $\mathtt{PAKE}$ *based on the concurrent 2PC is complete and simulatable in the plain model.*

For the proof we refer to Appendix B.

4.2 Concurrent Blind Signature in the Plain Model

Chaum [Cha82] introduced the notion of blind signatures (see Appendix A.2) and provided a concrete instantiation, while also showing an application to e-cash systems. Blind signatures have been in use as a building block for various other privacy-preserving crypto-systems such as e-voting [FOO92,Cha88], anonymous credential [CL01], and direct anonymous attestation [BCC04].

Blind signatures enable users to obtain a signature without revealing a message to be signed to a signer. More precisely, a blind signature scheme is a two-party computation between a signer and a user. The

signer has a pair of keys called verification-key and signing-key, and the user takes as input a message and the verification-key. They interact with each other, and the user obtains a signature for the message after the interaction. Definitions of security for blind signature schemes were first proposed by Pointcheval and Stern [PS00] and later there has been some refinements to the original definition [Fis06,Lin08,HKKL07]. At a high level, all existing definitions impose two basic security requirements on blind signatures: (i) users cannot forge a signature for a new message (unforgeability), and (ii) the signer cannot obtain information about the signed messages (blindness). For formal definitions see Appendix A.2.

When defining blindness and unforgeability it is crucial to distinguish whether security requires different executions of the protocol to be carried out sequentially (i.e., waiting for one execution to finish before beginning the next), or whether security holds even when multiple executions are performed concurrently (i.e., in an arbitrarily-interleaved manner). Concurrency in the context of blindness has received little attention, both because the 'standard' definition of blindness considers only two executions of the protocol and, because many known constructions of blind signature schemes achieve perfect blindness. In contrast, handling concurrency in the context of unforgeability has received much attention, and it is not hard to see that assuming there exist blind signature schemes at all there exist schemes that are unforgeable in the sequential setting but not in a concurrent setting.

Round-complexity and concurrent setting. One of the main performance measures for blind signatures is round-complexity. Let us call a round-optimal blind signature is a blind signature with only 2-moves, where the user and signer sends one message to each other. Another advantage is that round-optimal blind signatures are automatically secure in the concurrent setting [Lin08,HKKL07]

Before our work, constructing a blind signature that satisfies both round-optimal and concurrent security in the plain model was left an open problem. Using our concurrently secure 2PC, we present the first round-optimal concurrently-secure blind signature scheme that does not rely on random oracles or any setup assumptions such as a common reference string.

UC functionality of blind signatures. UC security is a framework proposed by Canetti [Can01] as a way to define security for protocols such that security-preserving composition is possible. Since UC security is a powerful and useful notion, an interesting question is how it relates to conventional security notions. In this line of the research, Fischlin [Fis06] investigated the context UC security of blind signatures in the CRS model and defined a ideal functionality in such setting. Later, Buan *et al.* [BGK06] proposed a new blind signature functionality in the plain model. But Buan *et al.*'s ideal functionality and its security definition only guarantees a version of blindness as weak blindness, reflecting the fact that the adversary is not allowed to choose his target keys. Also, their functionality requires the signer to be honest during key generation. But still defining a UC functionality of blind signature which guarantees standard blindness in the plain model, left as an open problem.

In this section, we define a new UC functionality of blind signature in Figure 3 that guarantees standard blindness. We will use it in the security proof of our concurrent blind signature scheme in Theorem 3. We note that, our blind signature construction enables one to achieve standard blindness, where the blindness holds for any adversarial choice of the keys as we let our 2PC scheme checking whether the key was indeed well-formed. We note that it is obvious that as our functionality for "concurrent" blind signature in Figure 3 guarantees blind signature properties (see the definition in Appendix A.2).

Concurrent blind signature construction. Our concurrent blind signature scheme builds on the concurrent two-party computation (2PC) in Figure 1. At a high level, we run only a part of the concurrent 2PC in Figure 1 in a way that one party only plays as receiver (P_{R}) and the other party plays as sender (P_{S}). And since for the application in a concurrent blind signature, we aim only P_{R} to compute the output of 2PC, $\mathtt{f}(\mathtt{x}_{\mathsf{R}}, \mathtt{x}_{\mathsf{S}})$, we need the SPSS.ZK proof $\pi_{\mathtt{zk}}$ only for $\mathcal{L}_2$ (instead of AND of $\mathcal{L}_2, \mathcal{L}_3$, see Section 3.2) in original 2PC in Figure 1. For the sake of completeness, we describe our concurrent blind signature scheme in Figure 4.

Building blocks. We construct a round-optimal concurrent blind signature scheme based on the following building blocks.

$\mathcal{F}_{\mathsf{BS}}$**:** It is parameterized by a message space $\mathcal{M}$, interacts with adversary Sim and a signer S, and parties $\mathsf{U}_1, \ldots, \mathsf{U}_n$ as follows.

- Upon receiving $(\mathsf{Gen}, \mathsf{sid}, \mathsf{S})$ from a signer S, proceed as follows:
  - if the signer S is honest, then inform Sim through $(\mathsf{Gen}, \mathsf{sid}, \mathsf{S})$ that a signature key generating request takes place. Then generate the signature key $\vec{k} := (\mathtt{BS.pk}, \mathtt{BS.sk})$ and output $(\mathsf{Key}, \mathsf{sid}, \mathsf{S}, \vec{k})$ to S.
  - If the signer S is corrupt, send $(\mathsf{Gen}, \mathsf{sid}, \mathsf{S})$ to Sim to obtain $(\mathsf{Key}, \mathsf{sidS}, \vec{k})$. Send $(\mathsf{Key}, \mathsf{sid}, \mathsf{S}, \vec{k})$ to S.
  - In either case record $(\mathsf{sid}, \mathsf{S}, \vec{k})$.
- Upon receiving $(\mathsf{Sign}, \mathsf{sid}, \mathsf{S}, \mathsf{U}_i, M)$ from U_i, where $M \in \mathcal{M}$, if a tuple $(\mathsf{sid}, \ldots)$ with the same sid was previously recorded, do nothing. Otherwise record $(\mathsf{sid}, \mathsf{S}, \mathsf{U}_i, M)$ and proceed as follows:
  - If the user U_i is honest, then inform S and Sim through $(\mathsf{Signature}, \mathsf{sid}, \mathsf{S}, \mathsf{U}_i)$ that a signature request takes place and then generate the signature σ and output $(\mathsf{Signature}, \mathsf{sid}, \mathsf{S}, \mathsf{U}_i, \sigma)$ to U_i (blindness condition).
  - If the user U_i is corrupt, then send $(\mathsf{Sign}, \mathsf{sid}, \mathsf{S}, \mathsf{U}_i, M)$ to Sim to obtain $(\mathsf{Signature}, \mathsf{sid}, \mathsf{S}, \mathsf{U}_i, \sigma)$. Send $(\mathsf{Signature}, \mathsf{sid}, \mathsf{S}, \mathsf{U}_i, \sigma)$ to U_i.
  - In either case record $(\mathsf{sid}, \mathsf{S}, \mathsf{U}_i, M, \sigma)$.
- Upon receiving $(\mathsf{Verify}, \mathsf{sid}, M, \sigma, \mathtt{BS.pk}')$ from some party P hand it to Sim and proceed a follows:
  - if $(\mathsf{sid}, \mathtt{BS.pk})$ and the tuple $(\mathsf{sid}, M, \sigma)$ were previously recorded then set $b = 1$ (completeness condition).
  - if $(\mathsf{sid}, \mathtt{BS.pk})$ was recorded but the tuple $(\mathsf{sid}, M, \sigma)$ was not previously recorded then set $b = 0$ (unforgeability condition).
  - send $(\mathsf{Verify}, \mathsf{sid}, M, \sigma, b)$ to the party P.

Fig. 3. Ideal functionality $\mathcal{F}_{\mathsf{BS}}$.

-Digital signature. Let $\mathsf{Sig} = (\mathsf{Sig.Gen}, \mathsf{Sig.S}, \mathsf{Sig.ver})$ is a digital signature scheme that is EUF-CMA against PPT adversaries. We assume that Sig is deterministic. This can be assumed without loss of generality by derandomizing the signing algorithm by using a secure PRF.

-Concurrent two-party computation. Let $\mathtt{c2PC} = (\mathtt{c2PC}.P_{\mathsf{R}}, \mathtt{c2PC}.P_{\mathsf{S}}, \mathtt{c2PC.ver})$ is the concurrent the concurrent two-party computation in Figure 1.

We also assume that the function $\mathtt{f}(\cdot, \cdot)$ used the concurrent 2PC in Figure 1 be a digital signature with the signing key $\mathsf{Sig.sk} := \mathtt{x}_{\mathsf{S}}$ and outputs the signature $\sigma \leftarrow \mathtt{f}(m, \mathsf{Sig.sk})$ on some message m.

Remark 1. We note that, in the blind signature application, one can use a two-round $\mathtt{CCA}$-secure commitment with reusable first message in the $\mathtt{c2PC}$, by simply setting the first message of the $\mathtt{CCA}$ commitment as part of the verification key.

We now give a more complete description of the protocol, along with the proof sketch of blindness and unforgeability.

Construction. Our round-optimal concurrent blind signature scheme $\mathtt{BS} = (\mathtt{BS.Gen}, \mathtt{BS.U}, \mathtt{BS.S}, \mathtt{BS.U}_{\mathsf{der}}, \mathtt{BS.ver})$ is described in Figure 4.

Security Proof We provide the proofs of blindness and unforgeability of Theorem 3 as its proof mainly relies on the security proof of the concurrent 2PC scheme in Theorem 1. For the sake of completeness, we write out the proof in their entirety. The correctness of the scheme immediately follows from the correctness of the digital signature Sig and the concurrent two-party computation $\mathtt{c2PC}$.

Theorem 3. *Assume two rounds concurrent 2PC protocol* $\mathtt{c2PC}$ *that is secure against malicious adversaries in the plain model. Assume a digital signature scheme* Sig *that is EUF-CMA against PPT adversaries. Then the concurrent blind signature* $\mathtt{BS}$ *is securely realizes* $\mathcal{F}_{\mathsf{BS}}$ *in the plain model.*

For the proof we refer to Appendix C.

Inputs: The party P_{S} has $\mathtt{BS.S}$'s input and the party P_{R} has the input of $\mathtt{BS.U}$

- **Round 1:**
  P_{R} Runs $\mathtt{BS.U}(\mathtt{BS.pk}, m := \mathtt{x_R})$ as follows:
  - $r_{\mathsf{R}} \leftarrow\$ \mathbb{Z}_p$ and $(\mathtt{FHE.pk}, \mathtt{FHE.sk}) \leftarrow \mathtt{FHE.Gen}(1^\lambda)$
  - $\mathtt{c_R} \leftarrow \mathtt{FHE.Enc}(\mathtt{FHE.pk_R}, \mathtt{x_R}; r_{\mathsf{R}})$
  - $\mathtt{cm_R} \leftarrow \mathtt{CCA.Com}(\mathtt{x_R}, r_{\mathsf{R}})$
  - $(\mathtt{ct_R}, k) \leftarrow \mathtt{CDS.R}(\mathtt{c_R}, \mathtt{w_R})$
  - $(\mathtt{ver_R}, \mathtt{zkst_R}) \leftarrow \mathtt{ZK}(1^\lambda)$

  P_{R} sends $(\mathtt{c_R}, \mathtt{cm_R}, \mathtt{ct_R}, \mathtt{ver_R}, \mathtt{FHE.pk})$
- **Round 2:**
  P_{S} runs $\mathtt{BS.S}(\mathtt{BS.sk}, \mathtt{BS.pk}, \mathcal{T}_1)$ as follows:
  - $r_{\mathsf{S}} \leftarrow\$ \mathbb{Z}_p$
  - $\mathtt{c_S} \leftarrow \mathtt{FHE.Eval}(\mathtt{FHE.pk}, \mathtt{c_R}, \mathtt{Sig.S}(\mathtt{Sig.sk}, \cdot); r_{\mathsf{S}})$
  - $\mathtt{cm_S} \leftarrow \mathtt{CCA.Com}(\mathtt{x_S}, r_{\mathsf{S}})$
  - $\mathtt{ct_S} \leftarrow \mathtt{CDS.S}(\mathtt{c_R}, \mathtt{c_S}, \mathtt{ct_R})$
  - $\pi_{\mathsf{zk}} \leftarrow \mathtt{ZK.P}(\mathtt{ver}, \mathtt{cm_S}, \mathtt{ct_S}, \mathtt{w}_i)$
  - P_{S} sends $\mathtt{cm_S}, \mathtt{ct_S}, \pi_{\mathsf{zk}}$
- **Signature derivation**
  P_{R} runs $\mathtt{BS.U_{der}}(\mathtt{FHE.sk}, \mathtt{zkst_R}, \mathcal{T}_2)$ as follows: if $\mathtt{ZK.V}(\mathtt{zkst}, \pi_{\mathsf{zk}}) = 1$ returns $\mathtt{FHE.Dec}(\mathtt{sk}, \mathtt{CDS.D}(k, \mathtt{ct_S}))$.
- **Signature verification**
  A party P runs $\mathtt{BS.ver}(\mathtt{BS.pk}, m, \mathtt{f})$ and outputs $\mathtt{Sig.ver}(\mathtt{BS.pk}, m, \mathtt{f})$.

Fig. 4. Generic construction of concurrent blind signature.

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 was shown in [BCKM20] that any classical 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 [BCKM20] 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 4. *Assuming the quantum sub-exponential hardness of the LWE problem and the quantum sub-exponentially hard* CCA*-secure commitments, there exists a two-round concurrent 2PC for all quantum circuits in the plain model.*

While the (sub-exponential) quantum hardness of the LWE problem is standard, we remark that (sub-exponential) quantum hardness of CCA-secure commitment requires us to make strong assumptions, such as quantum secure adaptive one-way functions [PPV08], or post-quantum indistinguishability obfuscation, for which a handful of candidates are known [GP20,WW20,BDGM20].

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 [BCKM20], except that we substitute our (classical) 2PC protocol from Figure 1 and we only require one party to know the output. Hence we drop the third round from the protocol described in [BCKM20].

The following section assumes familiarity with basic notions of quantum computation and quantum garbled circuits [BY20]. For a comprehensive introduction, we refer the reader to [BCKM20]. We are going to use the following schemes in our protocol:

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

We require that $\mathsf{poly}(\lambda) \ll \mathsf{T}_{\mathtt{c2PC}} \ll \mathsf{T}_{\mathtt{QGC}}$. In our protocol, parties P_0 and P_1 each having inputs $\mathbf{x}_0$ and $\mathbf{x}_1$ want to compute a quantum circuit Q on their inputs and get outputs $(\text{out}_1, \text{out}_2)$ 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 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 parties jointly encode their quantum inputs in two rounds and in parallel, they run a $\mathtt{c2PC}$ that outputs classical description of a quantum garbled circuit to the party who later will receive the output of the computation (lets assume P_2), then P_2 can get the output by evaluating the garbled circuit. More precisely, the classical functionality that our $\mathtt{c2PC}$ computes is called $\mathtt{f}[Q]$ and takes the classical description of Clifford unitaries $C_{1,in}, C_{1,out} \in \mathfrak{C}$ from P_1 and Clifford unitary $C_{2,in} \in \mathfrak{C}$ from P_2. Now imagine Q_1 is a quantum circuit that outputs $(C_{1,\text{out}}(\text{out}_1, \mathbf{0}^\lambda), \text{out}_2)$, it is clear to see that P_2 will be able to evaluate a garbling of Q_1 on a garbling of their joint inputs and get the output without learning P_1's output. The functionality $\mathtt{f}[Q]$ computes a garbling $(E, \hat{Q_1})$ of Q_1 where E is an input garbling Clifford. It computes

$$W = E \cdot (\mathbb{I} \otimes C_{2,in}^{-1} \otimes \mathbb{I}) \cdot C_{1,in}^{-1}$$

and finally, it outputs $(W, \hat{Q_1})$ to P_2 which later enables P_2 to compute their own output in the following way:

$$W(\mathbf{m}_1) = E \cdot (\mathbb{I} \otimes C_{2,in}^{-1} \otimes \mathbb{I}) \cdot C_{1,in}^{-1} \cdot C_{1,in}^{-1}(C_{1,in}(\mathbf{x}_1, \mathbf{m}_2, \mathbf{T}^{(k+1)\lambda}, \mathbf{0}^{k+\lambda}) = E(\mathbf{x}_1, \mathbf{x}_2, \mathbf{T}^{(k+1)\lambda}, \mathbf{0}^{k+\lambda})$$

Where $E(\mathbf{x}_1, \mathbf{x}_2, \mathbf{T}^k, \mathbf{0}^{k+\lambda})$ corresponds to the labels of the garbled circuit $\hat{Q_1}$, so P_2 can evaluate this circuit with the labels he and gets their output out_2. There are some caveats with this protocol as it is described above, and we will briefly mention them here and for more details we refer the reader to check [BCKM20].

This protocol is secure against a malicious P_2 but not against a malicious P_1 as they can put arbitrary $\mathbf{T}$ and $\mathbf{0}$s in the second round. To make the protocol secure against a malicious P_1 we use the "cut and choose" technique from [DGJ+20] and we refer to [BCKM20, §2,5] where they explained how this technique would be applied on their two-party protocol to make it secure against malicious Alice. The cut-and-choose technique is done by the Clifford unitary $\text{U}_{\text{dec-check-enc}}$ in the functionality of our $\mathtt{c2PC}$ that is used in the computation phase of our protocol below.

Now we describe our protocol in more detail:

- **Round 1** P_2 does the following:
  - Samples a random Clifford $C_{2,in}$ and uses it to encrypt and authenticate his input $\mathbf{x}_2$ as $\mathbf{m}_2 \leftarrow C_{2,in}(\mathbf{x}_2, \mathbf{0}^\lambda)$.
  - Computes the first round message m_2 of the $\mathtt{c2PC}$ using $C_{2,in}$ as their input.
  - Sends $(m_2, \mathbf{m}_2)$ to P_1.
- **Round 2** P_1 does the following:
  - Samples random Cliffords $C_{1,in}, C_{1,out}$ uses $C_{1,in}$ to encrypt and authenticate their own input $\mathbf{x}_1$ alongside with P_2's encoding $\mathbf{m}_2$.
  - Samples k copies of $\mathbf{T}$ state and $(k+1)\lambda$ copies of $\mathbf{0}$ sate.
  - Computes $\mathbf{m}_1 \leftarrow C_{1,in}(\mathbf{x}_1, \mathbf{m}_2, \mathbf{T}^{(k+1)\lambda}, \mathbf{0}^{k+\lambda})$.
  - Computes the second round message m_1 for the classical $\mathtt{c2PC}$ computation using $\mathtt{out} := (C_{1,in}, C_{1,out})$ as their input.
  - Sends $(m_1, \mathbf{m}_1)$ to P_2.
- **Computation Phase** P_2 does the following computation:
  - Using m_1 they can compute the output of the classical $\mathtt{c2PC}$ that is $(\text{U}_{\text{dec-check-enc}}, E_0, \hat{Q_1})$.[5]

[5] The full description of the $\text{U}_{\text{dec-check-enc}}$ unitary is specified in [BCKM20, §5].

- Compute $(\mathbf{m}_{\text{inp}}, \mathbf{z}_{\text{check}}, \text{trap}_2, \text{T}_{\text{inp}}, \mathbf{t}_{\text{check}}) \leftarrow \text{U}_{\text{dec-check-enc}}(\mathbf{m}_1)$
- 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 measurment is not zero.
- Compute $\text{out}_2 \leftarrow \texttt{QGEval}(E_0, \hat{Q}_1, \mathbf{m}_{\text{inp}})$.

5.2 Security Proof of Quantum Concurrent 2PC

In this section, we prove Lemma 5 that is similar to the proof of Theorem 5.1 in [BCKM20]. For the sake of completeness, we write out the proof in its entirety.

Lemma 5. *Assuming two-round concurrently secure 2PC protocol with black-box super-polynomial simulation and sub-exponentially secure quantum garbled circuits, 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 two cases when (i) a quantum PT adversary $\mathcal{A}$ corrupting party P_1 (ii) and $\mathcal{A}$ corrupting party P_2. For the ideal functionality, we use the Classical functionality in Figure 1 of [BCKM20].

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

– Compute $m_2 \leftarrow \mathsf{c2PC.Sim}(1^\lambda)$, samples a random Clifford $C_{2,in}$, and compute $\mathbf{m}_2 := C_{2,in}(\vec{0}, \mathbf{0}^\lambda)$. Send $(m_2, \mathbf{m}_2)$ to $\mathcal{A}(\mathbf{x}_1, \mathsf{aux}_{\mathcal{A}})$.

– Receive $(m_1, \mathbf{m}_1)$ from $\mathcal{A}$ and compute $\mathsf{out} \leftarrow \mathsf{c2PC.Sim}(1^\lambda, m_1)$. Abort if $\mathsf{out} = \perp$, otherwise, parse out as $(C_{1,in}, C_{1,out})$

– Sample $\text{U}_{\text{dec-check-enc}}$ by using $(C_{1,in}, C_{2,in})$ and compute $(\mathbf{x}'_1, \mathbf{x}'_2, \mathbf{m}_{\text{inp}}, \text{trap}_2 \mathbf{z}_{\text{check}}, \text{trap}_1, \text{T}_{\text{inp},\mathbf{t}_{\text{check}}}) \leftarrow \text{U}_{\text{dec-check-enc}}(\mathbf{m}_1)$. 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.

Case 2: Consider any quantum PT adversary $\mathcal{A}$ corrupting party P_2. The simulator $\mathsf{Sim}(\mathbf{x}_2, \mathsf{aux}_{\mathcal{A}})$ is defined as follows:

– Receive $(m_2, \mathbf{m}_2)$ from $\mathcal{A}$ and compute $\mathsf{inp} \leftarrow \mathsf{c2PC.Sim}(1^\lambda, m_2)$. If $\mathsf{inp} = \perp$ the abort, else parse inp as $C_{2,in}$ and compute $(\mathbf{x}'_2, \text{trap}_2) := C^\dagger_{2,in}(\mathbf{m}_2)$.

– Query ideal functionality and compute simulated round 2 message as follows:
  - Compute $\hat{\mathbf{m}}_{\text{inp}}$ by running $\texttt{QGSim}$ where where $\hat{\mathbf{m}}_{\text{inp}}$ is the simulated quantum garbled.
  - Sample a random $\text{U}_{\text{dec-check-enc}}$ and compute $\mathbf{m}_1 :=\leftarrow \text{U}^\dagger_{\text{dec-check-enc}}(\hat{\mathbf{m}}_{\text{inp}}, \vec{0}, \text{trap}_2, \text{T})$.
  - Compute $m_1 \leftarrow \mathsf{c2PC.Sim}(1^\lambda, \text{U}_{\text{dec-check-enc}})$.
  - Send $(m_1, \mathbf{m}_1)$ to $\mathcal{A}$.

Intuitively, 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 games where the first game Game_0 corresponds to the real world and the last game corresponds to the ideal world.

1. Game_1: In this game, Sim runs c2PC.Sim and simulates the c2PC scheme, using c2PC.Sim to extract $\mathcal{A}$'s input $C_{2,in}$, and c2PC.Sim to sample party P_1 message m_1. Use $C_{2,in}$ and freshly sampled ($C_{1,in}$ to sample the output of the classical functionality that is given to c2PC.Sim.
   The (computational) indistinguishability of Game_0 and Game_1 comes directly from the security against corrupted c2PC scheme.

2. Game_2: Now, we make a (perfectly indistinguishable) switch in how $\mathbf{m}_1$ is computed and how $\mathrm{U}_{\text{dec-check-enc}}$ (part of the classical 2c2PC output) is sampled. Define $(\mathbf{x}'_2, \mathrm{trap}_2) := C^{\dagger}_{2,in}(\mathbf{m}_2)$, where $C_{2,in}$ was extracted from m_2. As here exists a Clifford unitary U such that $\mathrm{U}_{\text{dec-check-enc}} = UC^{\dagger}_{1,in}$, where $C_{1,in}$ 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 $\mathbf{m}_1 := \mathrm{U}^{\dagger}_{\text{dec-check-enc}}(E_0, \vec{0}, \mathrm{trap}_2, \mathrm{T})$. This is how Game_2 is defined. Notice that, in Game_1, we have that, $\mathrm{U}_{\text{dec-check-enc}}(\mathbf{m}_1) := (E_0, \vec{0}, \mathrm{trap}_2, \mathrm{T})$.
   We observe that Game_1 and Game_2 are equivalent.

3. Game_3: In this game, we simulate the quantum garbled circuit. In particular, compute $(\hat{\mathbf{y}}_1, \mathbf{y}_2) \leftarrow \mathsf{Q}_{\mathsf{dist}}[C_{\text{out}}](\mathbf{x}_1, \mathbf{x}'_2, \vec{0}, \mathrm{T})$ and the compute $\hat{\mathbf{m}}_{\text{inp}}$ by running QGSim and the substitute $\hat{\mathbf{m}}_{\text{inp}}$ for E_0 in the computation of $\mathbf{m}_1$, so that $\mathbf{m}_1 := \mathrm{U}^{\dagger}_{\text{dec-check-enc}}(E_0, \vec{0}, \mathrm{trap}_2, \mathrm{T})$.
   The (computational) indistinguishability of Game_2 and Game_3 comes directly from the sub-exponential security of the QGC.

4. Game_4: We notice that $\mathsf{Q}_{\mathsf{dist}}$ may be computed in two stages, where the first outputs $(\mathbf{y}_1, \mathbf{y}_2, \vec{0}, C_{\text{out}})$ and the second outputs $(C_{\text{out}}(\mathbf{y}_1, \vec{0}), \mathbf{y}_2)$. In this game, we compute only the first stage and set $\mathbf{y}_1$ aside and re-define the final output to be $(\hat{\mathbf{y}}_1, \mathbf{y}_2) := (C_{\text{out}}(\vec{0}), \mathbf{y}_2)$.
   The (statistical) indistinguishability of Game_3 and Game_4 follows directly from the (perfect) hiding and (statistical) authentication of the Clifford code.

5. Game_5: Finally, instead of directly computing $\mathbf{y}_2$ from the first stage of $\mathsf{Q}_{\mathsf{dist}}$, query the ideal functionality with $\mathbf{x}'_2$ and receive back $\mathbf{y}_2$. Now, during party P_1'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.
   We observe that Game_1 and Game_2 are equivalent .

□

References

AIR01. William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In Birgit Pfitzmann, editor, Advances in Cryptology – EUROCRYPT 2001, volume 2045 of Lecture Notes in Computer Science, pages 119–135, Innsbruck, Austria, May 6–10, 2001. Springer, Heidelberg, Germany.

AJ17. Prabhanjan Ananth and Abhishek Jain. On secure two-party computation in three rounds. In Yael Kalai and Leonid Reyzin, editors, TCC 2017: 15th Theory of Cryptography Conference, Part I, volume 10677 of Lecture Notes in Computer Science, pages 612–644, Baltimore, MD, USA, November 12–15, 2017. Springer, Heidelberg, Germany.

BCC04. Ernest F. Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, editors, ACM CCS 2004: 11th Conference on Computer and Communications Security, pages 132–145, Washington, DC, USA, October 25–29, 2004. ACM Press.

BCKM20. 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.

BD18. Zvika Brakerski and Nico Döttling. Two-message statistically sender-private ot from lwe. In Theory of Cryptography Conference, pages 370–390. Springer, 2018.

BDGM20. Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Candidate io from homomorphic encryption schemes. Cryptology ePrint Archive, Report 2020/394, 2020. https://ia.cr/2020/394.

BGJ+17. 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: 15th Theory of Cryptography Conference, Part I, volume 10677 of Lecture Notes in Computer Science, pages 743–775, Baltimore, MD, USA, November 12–15, 2017. Springer, Heidelberg, Germany.

BGK06. Aslak Bakke Buan, Kristian Gjøsteen, and Lillian Kråkmo. Universally composable blind signatures in the plain model. IACR Cryptol. ePrint Arch., 2006:405, 2006.

BL18. Fabrice Benhamouda and Huijia Lin. k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In Jesper Buus Nielsen and Vincent Rijmen, editors, Advances in Cryptology – EUROCRYPT 2018, Part II, volume 10821 of Lecture Notes in Computer Science, pages 500–532, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

BMP00. Victor Boyko, Philip D. MacKenzie, and Sarvar Patel. Provably secure password-authenticated key exchange using Diffie-Hellman. In Bart Preneel, editor, Advances in Cryptology – EUROCRYPT 2000, volume 1807 of Lecture Notes in Computer Science, pages 156–171, Bruges, Belgium, May 14–18, 2000. Springer, Heidelberg, Germany.

BP12. Nir Bitansky and Omer Paneth. Point obfuscation and 3-round zero-knowledge. In Ronald Cramer, editor, TCC 2012: 9th Theory of Cryptography Conference, volume 7194 of Lecture Notes in Computer Science, pages 190–208, Taormina, Sicily, Italy, March 19–21, 2012. Springer, Heidelberg, Germany.

BPR00. Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In Bart Preneel, editor, Advances in Cryptology – EUROCRYPT 2000, volume 1807 of Lecture Notes in Computer Science, pages 139–155, Bruges, Belgium, May 14–18, 2000. Springer, Heidelberg, Germany.

BS05. Boaz Barak and Amit Sahai. How to play almost any mental game over the net - concurrent composition via super-polynomial simulation. In 46th Annual Symposium on Foundations of Computer Science, pages 543–552, Pittsburgh, PA, USA, October 23–25, 2005. IEEE Computer Society Press.

BY20. Zvika Brakerski and Henry Yuen. Quantum garbled circuits, 2020.

Can01. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In 42nd Annual Symposium on Foundations of Computer Science, pages 136–145, Las Vegas, NV, USA, October 14–17, 2001. IEEE Computer Society Press.

Cha82. David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, Advances in Cryptology – CRYPTO'82, pages 199–203, Santa Barbara, CA, USA, 1982. Plenum Press, New York, USA.

Cha88. David Chaum. Elections with unconditionally-secret ballots and disruption equivalent to breaking RSA. In C. G. Günther, editor, Advances in Cryptology – EUROCRYPT'88, volume 330 of Lecture Notes in Computer Science, pages 177–182, Davos, Switzerland, May 25–27, 1988. Springer, Heidelberg, Germany.

CHK+05. Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip D. MacKenzie. Universally composable password-based key exchange. In Ronald Cramer, editor, Advances in Cryptology – EUROCRYPT 2005, volume 3494 of Lecture Notes in Computer Science, pages 404–421, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany.

CKL03. Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. In Eli Biham, editor, Advances in Cryptology – EUROCRYPT 2003, volume 2656 of Lecture Notes in Computer Science, pages 68–86, Warsaw, Poland, May 4–8, 2003. Springer, Heidelberg, Germany.

CL01. Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, Advances in Cryptology – EUROCRYPT 2001, volume 2045 of Lecture Notes in Computer Science, pages 93–118, Innsbruck, Austria, May 6–10, 2001. Springer, Heidelberg, Germany.

CLP10. Ran Canetti, Huijia Lin, and Rafael Pass. Adaptive hardness and composable security in the plain model from standard assumptions. In 51st Annual Symposium on Foundations of Computer Science, pages 541–550, Las Vegas, NV, USA, October 23–26, 2010. IEEE Computer Society Press.

DGI+19. Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In Alexandra Boldyreva and Daniele Micciancio, editors, Advances in Cryptology – CRYPTO 2019, Part III, volume 11694 of Lecture Notes in Computer Science, pages 3–32, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany.

DGJ+20. 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, Advances in Cryptology – EUROCRYPT 2020, Part III, volume 12107 of Lecture Notes in Computer Science, pages 729–758, Zagreb, Croatia, May 10–14, 2020. Springer, Heidelberg, Germany.

Fis06. Marc Fischlin. Round-optimal composable blind signatures in the common reference string model. In Cynthia Dwork, editor, Advances in Cryptology – CRYPTO 2006, volume 4117 of Lecture Notes in Computer Science, pages 60–77, Santa Barbara, CA, USA, August 20–24, 2006. Springer, Heidelberg, Germany.

FJK21. 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.

FOO92. Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In International Workshop on the Theory and Application of Cryptographic Techniques, pages 244–251. Springer, 1992.

FS90. Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In 22nd Annual ACM Symposium on Theory of Computing, pages 416–426, Baltimore, MD, USA, May 14–16, 1990. ACM Press.

GGJS12. Sanjam Garg, Vipul Goyal, Abhishek Jain, and Amit Sahai. Concurrently secure computation in constant rounds. In David Pointcheval and Thomas Johansson, editors, Advances in Cryptology – EUROCRYPT 2012, volume 7237 of Lecture Notes in Computer Science, pages 99–116, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

GJ13. Vipul Goyal and Abhishek Jain. On concurrently secure computation in the multiple ideal query model. In Thomas Johansson and Phong Q. Nguyen, editors, Advances in Cryptology – EUROCRYPT 2013, volume 7881 of Lecture Notes in Computer Science, pages 684–701, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany.

GKP17. 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, Advances in Cryptology – EUROCRYPT 2017, Part II, volume 10211 of Lecture Notes in Computer Science, pages 194–224, Paris, France, April 30 – May 4, 2017. Springer, Heidelberg, Germany.

GLP+15. 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: 12th Theory of Cryptography Conference, Part I, volume 9014 of Lecture Notes in Computer Science, pages 260–289, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany.

GMR06. Craig Gentry, Philip MacKenzie, and Zulfikar Ramzan. A method for making password-based key exchange resilient to server compromise. In Cynthia Dwork, editor, Advances in Cryptology – CRYPTO 2006, volume 4117 of Lecture Notes in Computer Science, pages 142–159, Santa Barbara, CA, USA, August 20–24, 2006. Springer, Heidelberg, Germany.

GMW87. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, 19th Annual ACM Symposium on Theory of Computing, pages 218–229, New York City, NY, USA, May 25–27, 1987. ACM Press.

GP20. Romain Gay and Rafael Pass. Indistinguishability obfuscation from circular security. Cryptology ePrint Archive, Report 2020/1010, 2020. https://ia.cr/2020/1010.

GS17. Sanjam Garg and Akshayaram Srinivasan. Garbled protocols and two-round MPC from bilinear maps. In Chris Umans, editor, 58th Annual Symposium on Foundations of Computer Science, pages 588–599, Berkeley, CA, USA, October 15–17, 2017. IEEE Computer Society Press.

GS18. Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, Advances in Cryptology – EUROCRYPT 2018, Part II, volume 10821 of Lecture Notes in Computer Science, pages 468–499, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

HJK+18. Jung Yeon Hwang, Stanislaw Jarecki, Taekyoung Kwon, Joohee Lee, Ji Sun Shin, and Jiayu Xu. Round-reduced modular construction of asymmetric password-authenticated key exchange. In Dario Catalano and Roberto De Prisco, editors, SCN 18: 11th International Conference on Security in Communication Networks, volume 11035 of Lecture Notes in Computer Science, pages 485–504, Amalfi, Italy, September 5–7, 2018. Springer, Heidelberg, Germany.

HK12. Shai Halevi and Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. Journal of Cryptology, 25(1):158–193, January 2012.

HKKL07. Carmit Hazay, Jonathan Katz, Chiu-Yuen Koo, and Yehuda Lindell. Concurrently-secure blind signatures without random oracles or setup assumptions. In Salil P. Vadhan, editor, TCC 2007: 4th Theory of Cryptography Conference, volume 4392 of Lecture Notes in Computer Science, pages 323–341, Amsterdam, The Netherlands, February 21–24, 2007. Springer, Heidelberg, Germany.

JKX18. Stanislaw Jarecki, Hugo Krawczyk, and Jiayu Xu. OPAQUE: An asymmetric PAKE protocol secure against pre-computation attacks. In Jesper Buus Nielsen and Vincent Rijmen, editors, Advances in Cryptology – EUROCRYPT 2018, Part III, volume 10822 of Lecture Notes in Computer Science, pages 456–486, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

JLS20. Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. Cryptology ePrint Archive, Report 2020/1003, 2020. https://ia.cr/2020/1003.

Khu21. Dakshita Khurana. Non-interactive distributional indistinguishability (nidi) and non-malleable commitments, 2021.

Kiy14. Susumu Kiyoshima. Round-efficient black-box construction of composable multi-party computation. In Juan A. Garay and Rosario Gennaro, editors, Advances in Cryptology – CRYPTO 2014, Part II, volume 8617 of Lecture Notes in Computer Science, pages 351–368, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.

KK19. Yael Tauman Kalai and Dakshita Khurana. Non-interactive non-malleability from quantum supremacy. In Alexandra Boldyreva and Daniele Micciancio, editors, Advances in Cryptology – CRYPTO 2019, Part III, volume 11694 of Lecture Notes in Computer Science, pages 552–582, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany.

KMO14. Susumu Kiyoshima, Yoshifumi Manabe, and Tatsuaki Okamoto. Constant-round black-box construction of composable multi-party computation protocol. In Yehuda Lindell, editor, TCC 2014: 11th Theory of Cryptography Conference, volume 8349 of Lecture Notes in Computer Science, pages 343–367, San Diego, CA, USA, February 24–26, 2014. Springer, Heidelberg, Germany.

KS17. Dakshita Khurana and Amit Sahai. How to achieve non-malleability in one or two rounds. Cryptology ePrint Archive, Report 2017/291, 2017. https://eprint.iacr.org/2017/291.

Lin08. Yehuda Lindell. Lower bounds and impossibility results for concurrent self composition. Journal of Cryptology, 21(2):200–249, April 2008.

LP07. Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In Moni Naor, editor, Advances in Cryptology – EUROCRYPT 2007, volume 4515 of Lecture Notes in Computer Science, pages 52–78, Barcelona, Spain, May 20–24, 2007. Springer, Heidelberg, Germany.

LP12. Huijia Lin and Rafael Pass. Black-box constructions of composable protocols without set-up. In Reihaneh Safavi-Naini and Ran Canetti, editors, Advances in Cryptology – CRYPTO 2012, volume 7417 of Lecture Notes in Computer Science, pages 461–478, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany.

LPS17. Huijia Lin, Rafael Pass, and Pratik Soni. Two-round and non-interactive concurrent non-malleable commitments from time-lock puzzles. In Chris Umans, editor, 58th Annual Symposium on Foundations of Computer Science, pages 576–587, Berkeley, CA, USA, October 15–17, 2017. IEEE Computer Society Press.

MMY06. Tal Malkin, Ryan Moriarty, and Nikolai Yakovenko. Generalized environmental security from number theoretic assumptions. In Shai Halevi and Tal Rabin, editors, TCC 2006: 3rd Theory of Cryptography Conference, volume 3876 of Lecture Notes in Computer Science, pages 343–359, New York, NY, USA, March 4–7, 2006. Springer, Heidelberg, Germany.

MPR06. Silvio Micali, Rafael Pass, and Alon Rosen. Input-indistinguishable computation. In 47th Annual Symposium on Foundations of Computer Science, pages 367–378, Berkeley, CA, USA, October 21–24, 2006. IEEE Computer Society Press.

NP01. Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In SODA, volume 1, pages 448–457, 2001.

Ode01. Goldreich Oded. Foundations of cryptography basic tools, 2001.

OPCPC14. Rafail Ostrovsky, Anat Paskin-Cherniavsky, and Beni Paskin-Cherniavsky. Maliciously circuit-private fhe. In Annual Cryptology Conference, pages 536–553. Springer, 2014.

OPP14. Rafail Ostrovsky, Anat Paskin-Cherniavsky, and Beni Paskin-Cherniavsky. Maliciously circuit-private FHE. In Juan A. Garay and Rosario Gennaro, editors, Advances in Cryptology – CRYPTO 2014, Part I, volume 8616 of Lecture Notes in Computer Science, pages 536–553, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.

Pas03a. Rafael Pass. On deniability in the common reference string and random oracle model. In Dan Boneh, editor, Advances in Cryptology – CRYPTO 2003, volume 2729 of Lecture Notes in Computer Science, pages 316–337, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany.

Pas03b. Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In Eli Biham, editor, Advances in Cryptology – EUROCRYPT 2003, volume 2656 of Lecture Notes in Computer Science, pages 160–176, Warsaw, Poland, May 4–8, 2003. Springer, Heidelberg, Germany.

Pas04a. Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In László Babai, editor, 36th Annual ACM Symposium on Theory of Computing, pages 232–241, Chicago, IL, USA, June 13–16, 2004. ACM Press.

Pas04b. Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In Proceedings of the thirty-sixth annual ACM symposium on Theory of computing, pages 232–241, 2004.

PLV12. Rafael Pass, Huijia Lin, and Muthuramakrishnan Venkitasubramaniam. A unified framework for UC from only OT. In Xiaoyun Wang and Kazue Sako, editors, Advances in Cryptology – ASIACRYPT 2012, volume 7658 of Lecture Notes in Computer Science, pages 699–717, Beijing, China, December 2–6, 2012. Springer, Heidelberg, Germany.

PPV08. Omkant Pandey, Rafael Pass, and Vinod Vaikuntanathan. Adaptive one-way functions and applications. In David Wagner, editor, Advances in Cryptology – CRYPTO 2008, volume 5157 of Lecture Notes in Computer Science, pages 57–74, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany.

PS00. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. Journal of Cryptology, 13(3):361–396, June 2000.

PS04. Manoj Prabhakaran and Amit Sahai. New notions of security: Achieving universal composability without trusted setup. In László Babai, editor, 36th Annual ACM Symposium on Theory of Computing, pages 242–251, Chicago, IL, USA, June 13–16, 2004. ACM Press.

WW20. Hoeteck Wee and Daniel Wichs. Candidate obfuscation via oblivious lwe sampling. Cryptology ePrint Archive, Report 2020/1042, 2020. `https://ia.cr/2020/1042`.

Yao86. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In 27th Annual Symposium on Foundations of Computer Science, pages 162–167, Toronto, Ontario, Canada, October 27–29, 1986. IEEE Computer Society Press.

A More Preliminaries

A.1 Universal Composability

The Universal Composability (UC) framework was introduced by Canetti [Can01]. In the UC framework, one analyzes the security of the protocol under the real-world and ideal-world paradigm. More precisely, in this setting, the real-world execution of a protocol is compared with an ideal-world interaction with the primitive that it implements. Then a composition theorem in this model states that the security of the UC-secure protocols remains if it is arbitrarily composed with other UC-secure protocols or the protocol itself. Additionally, the UC-secure property guarantees security in practical applications where individual instances of protocols are run in parallel, such as the internet. The entities in the UC framework in both ideal-word and real-word executions are modeled as PPT interactive Turing machines that send and receive messages through respectively their output and input tapes.

In the ideal world execution, dummy parties (possibly controlled by an ideal-world adversary Sim, also called simulator) communicate directly with the ideal functionality $\mathcal{F}$. The ideal functionality can be viewed as a trusted party that creates the primitives to implement the protocol. Correspondingly, in the real-world execution, parties (possibly corrupted by a real-world adversary $\mathcal{A}$) communicate with each other as a protocol Π that realizes the ideal functionality. Both the ideal and real executions are controlled by the environment $\mathcal{Z}$, an entity that sends inputs and receives the outputs of $\mathcal{A}$, the individual parties, and Sim. Finally, after seeing the ideal or real protocol execution, $\mathcal{Z}$ returns a bit, which is considered as the output of the execution. Then the rationale behind this framework lies in showing that the environment $\mathcal{Z}$ can not efficiently distinguish between the ideal and real executions, therefore meaning that the real-world protocol is as secure as the ideal-world (the ideal functionality).

Besides, the two aforementioned models (real-world and ideal-world) of computation, the UC framework considers the hybrid-world, where the executions are similar to the real-world but with the additional assumption that the parties are allowed to access to an auxiliary ideal functionality $\mathcal{F}$. More precisely, in this case, instead of honest parties interacting directly with the ideal functionality, the adversary passes all the messages from and to the ideal functionality. Also, the transmission channels are considered to be ideally authenticated, meaning that the adversary is not able to modify the messages but only able to read them. Unlike information transferred between parties, which can be read by the adversary, the information transferred between parties and the ideal functionality is split into a public and private header. The private header carries some information like as the private inputs of parties and it cannot be read by the adversary. The public header carries only some information that can be viewed publicly such as receiver, sender, type of a message, and session identifiers. Let denote the output of the environment $\mathcal{Z}$ that shows the execution of a protocol Π in a real-world model and a hybrid model, respectively as $\mathsf{IDEAL}^{\mathcal{F}}_{\mathsf{Sim}}$ and $\mathsf{HYBRID}^{\mathcal{F}}_{\Pi,\mathcal{A}}$. Then the UC security is formally defined as:

Definition 7. *A protocol Π UC-realizes an ideal functionality $\mathcal{F}$ in the hybrid model if, for every PPT adversary $\mathcal{A}$, there exists a simulator Sim such that for all environments $\mathcal{Z}$, $\mathsf{IDEAL}^{\mathcal{F}}_{\mathsf{Sim}} \approx \mathsf{HYBRID}^{\mathcal{F}}_{\Pi,\mathcal{A}}$. The protocol Π is statistically secure if the above definition holds for all unbounded $\mathcal{Z}$.*

UC Security with super-polynomial simulation We next recall the relaxed notion of UC security by giving the simulator access to super-poly computational resources [GGJS12].

Definition 8. *A protocol Π UC-realizes an ideal functionality $\mathcal{F}$ in the hybrid model if, for every a super-polynomial time adversary $\mathcal{A}$, there exists a simulator Sim such that for all environments $\mathcal{Z}$, $\mathsf{IDEAL}^{\mathcal{F}}_{\mathsf{Sim}} \approx \mathsf{HYBRID}^{\mathcal{F}}_{\Pi,\mathcal{A}}$. The protocol Π is statistically secure if the above definition holds for all unbounded $\mathcal{Z}$*

The universal composition theorem generalizes naturally to the case of UC-SPS, the details of which we refer [GGJS12].

Ideal functionality of two-party computation. As we mentioned before, the security of a protocol is analyzed by comparing what an adversary can do in the protocol to what it can do in an ideal scenario that

is secure by definition. This is formalized by considering an ideal computation involving an incorruptible trusted third party to whom the parties send their inputs. The trusted party computes the functionality on the inputs and returns to each party its respective output. Here, we briefly review the ideal functionality of two-party computation and for more details refer to [LP07,GGJS12]. An ideal execution proceeds as follows:
Inputs: each party obtains an input $\mathtt{x}$ ($\mathtt{x} = \mathtt{x}_1$ for P_1, and $\mathtt{x} = \mathtt{x}_1$ for P_2).
Send inputs to trusted party: an honest party always sends $\mathtt{x}$ to the trusted party. A malicious party may, depending on $\mathtt{x}$, either abort or send some $\mathtt{x}' \in \{0,1\}^{|\mathtt{x}|}$.
Trusted party answers first party: in case it has obtained an input pair $(\mathtt{x}_1, \mathtt{x}_2)$, the trusted party first replies to the first party with $\mathtt{f}(\mathtt{x}_1, \mathtt{x}_2)$. Otherwise (i.e., in case it receives only one valid input), the trusted party replies to both parties with a special symbol $\perp$ to the trusted party.
Trusted party answers second party: in case the first party is malicious it may, depending on its input and the trusted party's answer, decide to stop the trusted party by sending it $\perp$ after receiving its output. In this case the trusted party sends $\perp$ to the second party. Otherwise (i.e., if not stopped), the trusted party sends $\mathtt{f}(\mathtt{x}_1, \mathtt{x}_2)$ to the second party.
Outputs: an honest party always outputs the message it has obtained from the trusted party. A malicious party may output an arbitrary function of its initial input and the message obtained from the trusted party.

Informally, we say that a protocol is secure if any adversary interacting in the real protocol (where no trusted third party exists) can do no more attack than if it was involved in the above-described ideal computation.

A.2 Blind Signatures

Here, we give a definition of blind signatures. For simplicity, we give a definition focusing on round-optimal blind signatures.

Definition 9. *A round-optimal blind signature scheme with a message space $\mathcal{M}$ consists of PPT algorithms $\Pi.\mathtt{BS} = (\mathtt{BS.Gen}, \mathtt{BS.U}, \mathtt{BS.S}, \mathtt{BS.U_{der}}, \mathtt{BS.ver})$.*

- $\mathtt{BS.Gen}(1^\lambda)$*: The key generation algorithm takes the security parameter 1^λ as input, and outputs a public key $\mathtt{BS.pk}$ and a secret key $\mathtt{BS.pk}$).*
- $\mathtt{BS.U}(\mathtt{BS.pk}, m)$*: The user's first message generation algorithm takes as input a public key $\mathtt{BS.pk}$, a secret key $\mathtt{FHE.sk}$ and a message m, and outputs a first message $\mathcal{T}_1$ and a state $\mathsf{st_U}$.*
- $\mathtt{BS.S}(\mathtt{BS.sk}, \tau)$*: The signer's second message generation algorithm takes as input a signing key $\mathtt{BS.sk}$, the first message τ, and outputs a signature ρ.*
- $\mathtt{BS.U_{der}}(\rho, \mathsf{st_U})$*: The user's signature derivation algorithm takes as input the state $\mathsf{st_U}$, a signature ρ and outputs a signature σ.*
- $\mathtt{BS.ver}(\mathtt{BS.pk}, m, \sigma)$*: The verification algorithm takes as input a public key $\mathtt{BS.pk}$, a message m, a signature σ, and outputs 1 to indicate acceptance or 0 to indicate rejection.*

A blind signature must satisfy the following properties:

- ***Correctness.*** *For any $\lambda \in \mathsf{N}$ and $m \in \mathcal{M}$,*

$$\Pr\left[\begin{array}{l}(\mathtt{BS.sk}, \mathtt{BS.pk}) \leftarrow \mathtt{BS.Gen}(1^\lambda); \tau \leftarrow \mathtt{BS.U}(\mathtt{BS.pk}, m);\\ \rho \leftarrow \mathtt{BS.S}(\mathtt{BS.sk}, \tau)\sigma \leftarrow \mathtt{BS.U_{der}}(\rho, \mathsf{st_U}) :\\ 1 \leftarrow \mathtt{BS.ver}(\mathtt{BS.pk}, m, \sigma)\end{array}\right] = 0 \ .$$

- ***Unforgeability.*** *For any $q = \mathsf{poly}(\lambda)$ and PPT adversary $\mathcal{A}$ that makes at most q queries, we have*

$$\Pr\left[\begin{array}{l}(\mathtt{BS.sk}, \mathtt{BS.pk}) \leftarrow \mathtt{BS.Gen}(1^\lambda); \{(m, \sigma)\}_{i\in[q+1]} \leftarrow \mathcal{A}(\mathtt{BS.pk}) :\\ 1 \leftarrow \mathtt{BS.ver}(\mathtt{BS.pk}, m, \sigma) \wedge \{m_i\}_{i\in[q+1]} \text{ is pairwise distinct.}\end{array}\right] \approx 0 \ ,$$

where we say that $\{m_i\}_{i\in[q+1]}$ is pairwise distinct if we have $m_i \neq m_j$ for all $i \neq j$.

– ***Blindness.*** *For defining blindness, we consider the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.*
  - *Setup. The adversary $\mathcal{A}$ is given as input the security parameter λ, and sends a public key $\mathsf{BS.pk}$ and a pair of messages (m_0, m_1) to $\mathcal{C}$.*
  - *First message. The challenger $\mathcal{C}$ generates $(\tau_b, \mathsf{st}_{\mathsf{0U_b}})$ $\mathsf{getsBS.U}(\mathsf{BS.pk}, \mathsf{m_b})$ for each $b \in \{0,1\}$, picks $\mathsf{coin} \leftarrow\$ \{0,1\}$, and gives $(\tau_{\mathsf{coin}}, \tau_{1-\mathsf{coin}})$ to $\mathcal{A}$.*
  - *Second message. The adversary $\mathcal{A}$ sends $(\rho_{\mathsf{coin}}, \rho_{1-\mathsf{coin}})$ to $\mathcal{C}$.*
  - *Signature derivation. The challenger generates $\sigma_b \leftarrow \mathsf{BS.U_{der}}(\rho_b, \mathsf{st_{U_b}})$ for each $b \in \{0,1\}$. $\mathcal{C}$ gives (σ_0, σ_1) to $\mathcal{A}$.*
  - *Guess. $\mathcal{A}$ outputs its guess coin'. We say that $\mathcal{A}$ wins if $\mathsf{coin} = \mathsf{coin}'$. We say that a blind signature scheme satisfies blindness if for any PPT adversary $\mathcal{A}$, we have $\Pr[\mathcal{A}\ wins] \approx 0$.*

B Security Proof of Concurrently Secure PAKE

(i: Completeness). This is straightforward from the construction and follows the directly from the correctness of the concurrent 2PC in Figure 1.

(ii: Simulatability). It follows directly from Theorem 1 (two rounds concurrent 2PC protocol for any functionality $\mathtt{f}$ that is secure against malicious adversaries in the plain model).

C Security Proof of Concurrently Secure Blind Signature

We build the ideal-model adversary S by black-box simulation of A, relaying all communication between the environment Z and the (simulated) adversary A, and acting on behalf of the honest parties in this simulation. Algorithm S also corrupts a dummy party in the ideal model whenever A asks to corrupt the corresponding party in the simulation. By assumption this is only done before the execution starts. Intuitively, we construct an ideal-world adversary Sim that runs a black- box simulation of the real-world adversary $\mathcal{A}$ by simulating the protocol execution and relaying messages between $\mathcal{A}$ and the environment $\mathcal{Z}$. We have to show that for each adversary $\mathcal{A}$ attacking the real-world protocol there exist an ideal-model adversary Sim in the ideal world with dummy parties and functionality $\mathcal{F}_{\mathsf{BS}}$ such that no environment $\mathcal{Z}$ can distinguish whether it is facing an execution in the real world with $\mathcal{A}$ or one in the ideal world with Sim. Sim proceeds as follows in experiment IDEAL:

– Suppose that the adversary lets a corrupt signer in the black-box simulation initiate a protocol run with the honest user by sending values $\mathcal{T}_2 := (\mathtt{c_S}, \mathtt{cm_S}, \mathtt{ct_S}, \pi_{\mathsf{zk}})$ (the message in the second round of concurrent blind signature $\mathtt{BS}$ scheme). Then the simulator Sim recover $(\mathsf{Sig.sk}, r_{\mathsf{S}})$ from $\mathcal{T}_2$ (and aborts if it fails) and submits $(\mathsf{Gen}, \mathsf{sid}, \mathsf{S})$ on behalf of the signer to the ideal functionality. It immediately receives a request $(\mathsf{Gen}, \mathsf{sid}, \mathsf{S})$ from $\mathcal{F}_{\mathsf{BS}}$. To answer, Sim computes the signature keys $\vec{k} := (\mathsf{Sig.pk}, \mathsf{Sig.sk}) \leftarrow \mathsf{Sig.Gen}(1^\lambda)$ of the unforgeable signature scheme Sign and sends $(\mathsf{Key}, \mathsf{sid}, \mathsf{S}, \vec{k})$ for back to the functionality.
– If an honest signer requests signature keys $(\mathsf{Gen}, \mathsf{sid}, \mathsf{S})$ in the ideal model and waits to receive $(\mathsf{Key}, \mathsf{sid}, \mathsf{S}, \vec{k})$, generated by the functionality, then the ideal-model adversary Sim generates $\mathcal{T}_2 = (\mathtt{c_S}, \mathtt{cm_S}, \mathtt{ct_S}, \pi_{\mathsf{zk}})$ where $\mathtt{c_R}$ and $\mathtt{cm_R}$ are $\mathtt{FHE.Eval}$ and $\mathtt{CCA.Com}$ of 0 respectively, π_{zk} is generated by running the $\mathsf{Sim_{zk}}$, and lets the signer in the black-box simulation send these values $\mathcal{T}_2$.
– Suppose that the adversary lets a corrupt user in the black-box simulation initiate a protocol run with the honest signer by sending values $\mathcal{T}_1 = (\mathtt{c_R}, \mathtt{cm_R}, \mathtt{ct_R}, \mathsf{ver}, \mathsf{pk})$ (the message in the first round of concurrent blind signature $\mathtt{BS}$ scheme). Then the simulator Sim recover (m, r_{R}) from $\mathcal{T}_1$ (and aborts if it fails) and submits $(\mathsf{Sign}, \mathsf{sid}, \mathsf{S}, \mathsf{U}, m)$ on behalf of the user to the ideal functionality. It immediately receives a request $(\mathsf{signature}, \mathsf{sid}, \mathsf{S}, \mathsf{U}, m)$ from $\mathcal{F}_{\mathsf{BS}}$. To answer, Sim computes the signature $\sigma := \mathtt{f} \leftarrow \mathsf{Sig.S}(\mathsf{Sig.sk}, m)$ under the unforgeable signature scheme Sign and sends $(\mathsf{signature}, \mathsf{sid}, \mathsf{S}, \mathsf{U}, \sigma)$ for back to the functionality.

– If an honest user requests a signature $(\mathsf{Sign}, \mathsf{sid}, \mathsf{S}, \mathsf{U}, m)$ in the ideal model and waits to receive $(\mathsf{signature}, \mathsf{sid}, \mathsf{S}, \mathsf{U}, \sigma)$, generated by the functionality, then the ideal-model adversary Sim generates $\mathcal{T}_1 = (\mathtt{c}_\mathsf{R}, \mathtt{cm}_\mathsf{R}, \mathtt{ct}_\mathsf{R}, \mathsf{ver}, \mathsf{pk})$ where $\mathtt{c}_\mathsf{R}$ and $\mathtt{cm}_\mathsf{R}$ are $\mathtt{FHE.Enc}$ and $\mathtt{CCA.Com}$ of 0 respectively and lets the user in the black-box simulation send these values $\mathcal{T}_1$.

This gives a full description of the ideal-model simulator. We claim that the differences are undetectable for the environment $\mathcal{Z}$. This is proven through a sequence of games transforming an execution in the ideal-model scenario into one which is equal to the one of the actual protocol.
Game_1**:** In this game, consider a simulator $\mathsf{Sim}_{\mathsf{Game}}$ that plays the role of the honest parties. $\mathsf{Sim}_{\mathsf{Game}}$ runs in polynomial time.
Game_2**:** In this game, the simulator $\mathsf{Sim}_{\mathsf{Game}}$ also runs the "Input Extraction" phase and the "Abort" phase as in Figure 2. $\mathsf{Sim}_{\mathsf{Game}}$ runs in time $\mathsf{T}^{\mathsf{Brk}}_{\mathtt{CCA.Com}}$.

Lemma 6. *Let* $\mathtt{c2PC}$ *is a concurrent 2PC (assuming soundness of the SPSS.ZK argument system, binding of the non-interactive CCA-commitment scheme and correctness of the* $\mathtt{FHE}$ *scheme), then* Game_1 *is computationally indistinguishable from* Game_2.

Proof. The only difference between the two games is that in Game_2, $\mathsf{Sim}_{\mathsf{Game}}$ may output "Abort" which doesn't happen in Game_1. More specifically, in Game_2, "Abort" occurs if the event E (see Game_2 in the proof of Theorem 1) is true. The event E occurs if for any malicious party, it gives valid ZK proofs in round 2 but its protocol statement is not consistent with the values it committed to. Therefore, in order to prove the indistinguishability of the two games, we note that based on the binding property of the commitment scheme, the correctness of the $\mathtt{FHE}$ scheme and the soundness property of the SP.SSZK argument system, $\Pr[\text{Event E is true in } \mathsf{Game}_2] = \mathsf{negl}(\lambda)$ (see Sub-Lemma 1 of Theorem 1). □

Game_3**:** This game is identical to the previous game except that in Round 2, $\mathsf{Sim}_{\mathsf{Game}}$ now computes simulated SPSS.ZK proofs as done in Figure 2. $\mathsf{Sim}_{\mathsf{Game}}$ runs in time $\mathsf{T}^{\mathsf{Brk}}_{\mathtt{CCA.Com}}$.

Lemma 7. *Assuming the zero knowledge property of the SPSS.ZK argument system of* $\mathtt{c2PC}$ *scheme, then* Game_2 *is computationally indistinguishable from* Game_3.

Proof. The only difference between the two games is that in Game_2, $\mathsf{Sim}_{\mathsf{Game}}$ computes the proofs in Round 2 honestly, by running the algorithm $\mathtt{ZK}$ of the SPSS.ZK argument system, whereas in Game_3, a simulated proof is used. If the adversary $\mathcal{A}$ can distinguish between the two games, we can use $\mathcal{A}$ to design an algorithm $\mathcal{A}_{\mathsf{zk}}$ that breaks the zero knowledge property of the argument system. The proof is similar to the proof of Lemma 2 in Theorem 1. □

Game_4**:** This is identical to the previous game except that $\mathsf{Sim}_{\mathsf{Game}}$ now computes both $\mathtt{cm}_\mathsf{R}$ and $\hat{\mathtt{cm}}_\mathsf{S}$ as $\mathtt{CCA}$ commitment of 0. $\mathsf{Sim}_{\mathsf{Game}}$ runs in time $\mathsf{T}^{\mathsf{Brk}}_{\mathtt{CCA.Com}}$.

Lemma 8. *Let* $\mathtt{CCA.Com}$ *of* $\mathtt{c2PC}$ *scheme is a* $\mathtt{CCA}$*-secure commitment scheme, then* Game_3 *is computationally indistinguishable from* Game_4.

The proof is a straightforward reduction to computational hiding of the $\mathtt{CCA}$ commitment (see proof of Lemma 3 in Theorem 1).
Game_5**:** This game is identical to the previous game except that in Round 2, $\mathsf{Sim}_{\mathsf{Game}}$ now computes the messages of the protocol using the simulator algorithms $\mathtt{FHE.Sim}$ as done by Sim in the ideal world. Sim also instructs the ideal functionality to deliver outputs to the honest parties as done by Sim. This game is now same as the ideal world.

Lemma 9. *Assuming the security of* $\mathtt{FHE}$ *scheme of* $\mathtt{c2PC}$ *scheme,* Game_4 *is computationally indistinguishable from* Game_5.

Proof. The only difference between the two games is that in Game_4, $\mathsf{Sim}_{\mathsf{Game}}$ computes the messages of protocol in Figure 4 correctly using the honest parties inputs, whereas in Game_5 the corresponding messages are computed by running the simulator $\mathtt{FHE}.\mathsf{Sim}$ for $\mathtt{FHE}$. The proof is similar to the proof of Lemma 4 in Theorem 1). □

All the steps in the final game now are exactly as in an attack on the real protocol with adversary $\mathcal{A}$. Therefore, the environment's output in the ideal-model simulation and the real-world execution are indistinguishable.