Unifying communication paradigms in delegated quantum computing

Fabian Wiesner[1,2], Jens Eisert[2,3], and Anna Pappa[1]

[1] Electrical Engineering and Computer Science Department,
Technische Universität Berlin, 10587 Berlin, Germany
[2] Dahlem Center for Complex Quantum Systems,
Freie Universität Berlin, 14195 Berlin, Germany
[3] Fraunhofer Heinrich-Hertz Institute, 10587 Berlin, Germany

Abstract. Delegated quantum computing (DQC) allows clients with low quantum capabilities to outsource computations to a server hosting a quantum computer. This process is typically envisioned within the measurement-based quantum computing framework, as it naturally facilitates blindness of inputs and computation. Hence, the overall process of setting up and conducting the computation encompasses a sequence of three stages: preparing the qubits, entangling the qubits to obtain the resource state, and measuring the qubits to run the computation. There are two primary approaches to distributing these stages between the client and the server that impose different constraints on cryptographic techniques and experimental implementations. In the prepare-and-send setting, the client prepares the qubits and sends them to the server, while in the receive-and-measure setting, the client receives the qubits from the server and measures them. Although these settings have been extensively studied independently, their interrelation and whether setting-dependent theoretical constraints are inevitable remain unclear. By implementing the key components of most DQC protocols in the respective missing setting, we provide a method to build prospective protocols in both settings simultaneously and to translate existing protocols from one setting into the other.

Keywords: Delegated Quantum Computing, Quantum verification, Composable Cryptography

1 Introduction

The potential for achieving unparalleled computational power beyond what is available with classical computers renders quantum computers highly appealing to academic institutions and companies alike. However, they are costly, prone to errors, and demand specialized maintenance and cooling systems beyond the capacity of standard laboratories. *Delegated quantum computing* (DQC) addresses these challenges, at least in principle, on a meaningful level of abstraction: it enables clients with limited quantum resources to outsource quantum computations to a high-performance quantum server.

Two crucial properties distinguish DQC from classical cloud access as it is already used today. *Blindness* guarantees that the server cannot learn the input to the computation or the computation itself. *Verifiability* allows the client to verify the output of the computation. While the impossibility of classical remote state preparation [1] suggests that a purely classical client is likely not able to achieve blind DQC without additional assumptions, a client with minimal quantum capabilities can blindly delegate a quantum computation when the latter is instantiated in the *measurement-based quantum computing model* (MBQC) [28,2].

Two separate lines of research – *prepare-and-send* (PS) and *receive-and-measure* (RM) – emerged. Both follow the convention of MBQC to encode computations as sequences of measurement angles in the XY-plane for qubit measurements of the resource state. Since the outcomes of qubit measurements of entangled states are inherently random, some outcomes introduce unintended Pauli operations on the qubits adjacent to the measured one. These Pauli operations can be systematically propagated forward in the measurement sequence by adapting the angles using the *flow function* associated with the resource state. This technique eliminates the necessity to apply quantum gates during the measurement stage by deferring them to the output phase.

Despite following the same convention, PS and RM differ significantly in the distribution of the steps of an MBQC computation among the server and the clients. In the context of delegated quantum computing, an MBQC computation is a sequence of four steps: (i) preparing single qubits, (ii) entangling these qubits to obtain the resource state using controlled Z (CZ) gates, (iii) conducting the measurement sequence and (iv) applying the corrections on the outcome. While the server will always perform step (ii) in both settings, the clients' behavior changes: in PS, the clients have to perform (i) and (iv) while in RM, the clients conduct (iii) – instead of (i) – and (iv).

These different distributions of responsibilities imply a difference in the techniques used to achieve blindness and verifiability in each setting. Beyond the implications for cryptographic techniques used for the protocol, the choice of setting is also consequential for the technological implementation. For example, in PS the clients need to be able to prepare single-qubit states while in RM they have to perform measurements. The chosen setting also imposes constraints on the operations on the server's side, and while photons are likely to serve for encoding flying qubits, it is uncertain which technology will enable the server to perform quantum computations. Indeed, some first proof-of-concept implementations of DQC in PS leverage photonic quantum computing [11,25], in contrast to a recent implementation in RM, which utilizes solid-state qubits [29]. To maintain flexibility regarding the hardware platform that will be used for future quantum servers, it is imperative to develop cryptographic techniques for both communication settings of DQC.

1.1 Related Work

The PS and RM settings have been extensively, albeit separately, studied. The first protocol proposing delegated quantum computing using MBQC is the work of Broadbent et al. [4] in 2009. This PS protocol achieves perfect blindness by using random offsets for the communicated measurement angles that cancel out offsets at the preparation – a technique that is now the standard approach for blindness in PS. While [4] showed only stand-alone security of the protocol, Dunjko et al. [8] proved composable security in 2018. Many other works built upon this first protocol. A protocol by Fitzsimons and Kashefi [9], proposed in 2017, is one of the first efforts toward verifiability. This protocol utilizes in PS that Z-eigenstates are invariant under CZ and, hence, prevent the establishment of entanglement. This mechanism allows splitting a resource state into sections, of which some serve for the actual computation and other parts, called traps, to test the honesty of the server. Crucially, the server does not know the partition of the resource state since the client sends the qubits for the resource state. Kashefi and Wallden [17] optimized this approach and proposed a protocol with significantly lower verification overhead, and Leichtle et al. [18] leveraged the properties of BQP computations to provide fault-tolerant verification of such computations. However, the number of clients was also varied: Kashefi and Pappa [16] proposed one of the first protocols allowing multiple clients to delegate a joint computation to an untrusted server, achieving blindness against the server or a subset of clients but not all coalitions of clients and the server. A recent work [14] in PS combines multi-client settings and verification – although not trap-based – and achieves blindness and verifiability as long as a single client is honest and the computation has a classical output utilizing a majority vote. Parallel to these developments for PS, research in the RM seeting produced several protocols, as well. The first is the perfectly blind single-client protocol by Morimae and Fujii in 2013 [24]. Shortly after that, Morimae proposed the first verification protocol in RM [22], in which – similarly to [9] – the client introduces traps. However, since the client does not prepare the qubits, the traps are introduced by delegating a precomputation. A different approach to verification was used a year later in a protocol that utilizes stabilizer testing to verify the resource state with the caveat that the input of the computation can only be classical [12]. Building upon this, Morimae lifted this caveat in 2016 [23].

1.2 Our contribution

While two protocols from different settings can exhibit similarities at a higher level by achieving or even utilizing the same techniques, the two settings are not obviously equivalent. More specifically, the interrelation of the settings is unknown, which raises the question of whether everything implemented in one setting can be implemented in the other with the same level of security. In our work, we follow a modular approach using the *abstract cryptography framework* [21] and answer this question positively on a practical level by investigating the abstract building blocks used in modern DQC protocols. We highlight existing or immediate correspondences between implementations of these building blocks, identify gaps in implementing these building blocks in the two settings, and close these gaps by providing the respective missing implementations. In more detail, we introduce a direct usage of traps in RM by means of proposing a trap-based equivalent of [17], we translate verification based on stabilizer testing as used in [12], which seems native to RM, to PS, and finally, implement collective remote state preparation as proposed in [16] in RM. Closing these gaps enables the setting-agnostic development of protocols, which eventually bridges the theory and experimental implementation.

We first introduce the abstract cryptography framework in section 2. In section 3, we present our results, i.e. the missing implementations of the building blocks of modern DQC. In the first subsection (3.1), we present and prove the security of an RM version of [17], followed by the translation of verification by stabilizer testing to PS. In the second subsection (3.2), we introduce collective remote state preparation in RM. Finally, we conclude our work in section 4.

2 Abstract cryptography

In order to demonstrate the modularity of the various protocols, we will use a composable security framework, namely *abstract cryptography* (AC) [21] which is closely related to constructive cryptography [20] and follows similar ideas as *universal composability* (UC) [6] and *categorical composable cryptography* (CCC) [5]. While we do not formally introduce every component of AC and refer to Refs. [21] and [26], we provide a brief explanation of how one defines security using these components.

The goal in AC is to construct an ideal (target) resource from a real one. Assuming that there is an honest subset $\mathcal{H}$ of all parties $\mathcal{I}$, we achieve this by showing that the composition of the honest parties' protocols attached to the real resource is indistinguishable from the ideal resource, where a simulator acts on the interfaces of the dishonest parties.

Definition 1 (Secure construction). *Let* $\mathbf{R}_\sharp = (\mathbf{R},\sharp)$ *and* $\mathbf{S}_\flat = (\mathbf{S},\flat)$ *be two pairs, each consisting of a resource with interface set* $\mathcal{I}$ *and a filter. For a set of honest parties* $\mathcal{H}$*, a protocol* $\pi = \{\pi_i\}_{i\in\mathcal{I}}$ *securely constructs* $\mathbf{S}_\flat$ *out of* $\mathbf{R}_\sharp$ *within* ϵ*, if there exists a simulator* $\sigma_{\mathcal{I}\setminus\mathcal{H}}$ *such that*

$$(\pi_\mathcal{H}\circ\sharp_\mathcal{H})\mathbf{R} \approx_\epsilon \flat_\mathcal{H}\mathbf{S}\sigma_{\mathcal{I}\setminus\mathcal{H}}. \tag{1}$$

In this case, we write $\mathbf{R}_\sharp \xrightarrow[\mathcal{H}]{\pi,\,\epsilon} \mathbf{S}_\flat$.

It is easier to understand the above definition with an example; in Fig. 1 we show how Eq. (1) looks like when $\mathcal{I}=\{A,B,C,D\}$ and $\mathcal{H}=\{A,B\}$. Note that we allow for global simulators in contrast to the usual local simulators in AC. This is not an unusual assumption and has been used before, both in AC [7], as well as in other frameworks ([6,5]).

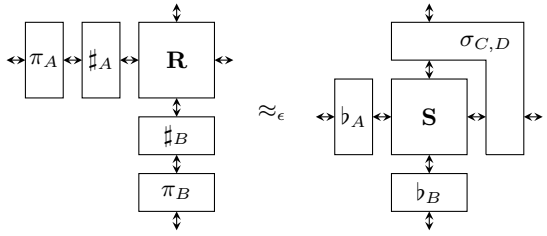

Fig. 1. Visualization of the secure construction with $\mathcal{I}=\{A,B,C,D\}$ and $\mathcal{H}=\{A,B\}$.

One shows security by proving the above definition for all sets of honest parties $\mathcal{H}\subseteq\mathcal{I}$ that are relevant for the security of the desired functionality. For example in QKD, it does not make sense to consider any of the two communicating parties to be dishonest; we are interested in proving a) correctness, where both the communicating parties and the eavesdropper are honest, and b) security, where the honest set contains only the communicating parties.

There are two reasons that motivate the definition of secure construction; first, any successful attack on the implementation implies an attack on the ideal resource which one finds by composing the attack with the simulator. Hence, we can encode in the ideal resource what should be possible for dishonest parties. As the access to the ideal resource is different depending on whether a party is honest or not, we apply filters for honest parties that block such additional interactions.

The second motivation is *composability*: as distinguishability is a pseudo-metric which is non-increasing under composition, we find for any filtered resources $\mathbf{R}_\sharp$,$\mathbf{S}_\flat$ and $\mathbf{Q}_\natural$ that

$$\mathbf{R}_\sharp \xrightarrow[\mathcal{H}]{\pi,\,\epsilon_1} \mathbf{S}_\flat \wedge \mathbf{S}_\flat \xrightarrow[\mathcal{H}]{\tau,\,\epsilon_2} \mathbf{Q}_\natural \Rightarrow \mathbf{R}_\sharp \xrightarrow[\mathcal{H}]{\tau\circ\pi,\,\epsilon_1+\epsilon_2} \mathbf{Q}_\natural. \tag{2}$$

3 Implementing the components for DQC

Essentially, all proposed MBQC-based DQC protocols include implementations of the following three components and leverage their properties:

1. Single-client blind DQC ($\mathbf{S}^{\mathrm{blind}}$).
2. Single-client verifiable DQC ($\mathbf{S}^{\mathrm{ver}}$).
3. Collective Remote State Preparation (RSP).

Interestingly, not all these components have been implemented in both the PS and the RM setting. In this work, we provide the missing implementations and therefore enable the execution of any type of protocol utilizing these components in either of the two settings.

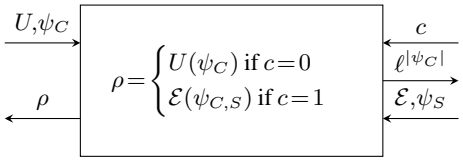

Fig. 2. *Visualization of* $\mathbf{S}^{\mathrm{blind}}$. $\ell^{|\psi_C|}$ is the size of the register for the client's input ψ_C, $\mathcal{E}$ is a completely positive trace-preserving (CPTP) map to the space of linear operators on $\mathbb{C}^{\ell^{|\psi_C|}}$, ψ_S is a register of the server and c denotes the server's behavior. If the server is honest, we assume a filter $\sharp_S$ which inputs $c=0$, ignores the received dimensionality of the client's state and inputs any CPTP map and register.

The first component, $\mathbf{S}^{\mathrm{blind}}$ (cf. Fig. 2), has already been perfectly implemented in both PS [4] and RM [24] as long as the client is honest.

In the following sections, we will therefore examine the two remaining components.

3.1 Single-client verifiable DQC

For the second component, $\mathbf{S}^{\mathrm{ver}}$ (cf. Fig.3), three types of implementations are used: the first is based on the cut-and-choose technique applied on $\mathbf{S}^{\mathrm{blind}}$ [15], i.e., repeated runs with trap rounds, the second type of implementation uses separated traps in the resource state [9,17] and the third leverages stabilizer measurements on the resource state [12].

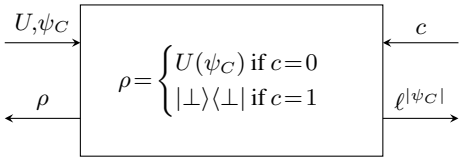

Fig. 3. *Visualization of* $\mathbf{S}^{\mathrm{ver}}$. $\ell^{|\psi_C|}$ is the size of the register for the client's input ψ_C. $|\perp\rangle\langle\perp|$ is orthogonal to the space of possible honest outputs. If the server is honest, we assume a filter $\flat_S$ which inputs $c=0$ and ignores the received dimensionality of the client's state.

The first type of implementation of $\mathbf{S}^{\mathrm{ver}}$ uses a composition of many instances of $\mathbf{S}^{\mathrm{blind}}$, where some of them are used as traps to check the behavior of the server. For protocols that use test rounds such as Ref. [3], one can treat $\mathbf{S}^{\mathrm{blind}}$ as a black box to build $\mathbf{S}^{\mathrm{ver}}$, and therefore the equivalence of the $\mathbf{S}^{\mathrm{blind}}$ implementations between the PS and the RM settings implies the equivalence of the cut-and-choose implementations of $\mathbf{S}^{\mathrm{ver}}$.

For the second type, we present in Protocol 1 an RM version of the protocol in Ref. [17]. We note that dummy nodes allow us to perform break operations, i.e., measuring a dummy deletes an edge. In this way, the traps are isolated, and the computational nodes are brought into the state of the dotted base state. The computation can be adapted so that one applies the bridge operation on the dots; this, together with the observation that for an honest server the isolated traps will be in $Z^{r_{N_{DT(G)}(t)}}|+\rangle$ and, hence, will not trigger abort, gives correctness.

Protocol 1 RM version of Ref. [17]

Input: Client inputs a quantum input ψ_C and secret measurement angles encoding U.
Output: $U(\psi_C)$ if the server is honest, $|\perp\rangle\langle\perp|$ otherwise.

1: Similar to Ref. [17], the client one-time pads the input state ψ_C and an additional register $|+\rangle\langle+|^{\otimes 2\ell^{|\psi_C|}}$. The combined state is denoted with $|e\rangle\langle e|$.
2: The server prepares the tripled-dotted resource state and entangles it with $|e\rangle\langle e|$, the resulting state is called ρ.
3: The client samples a coloring respecting the position of the input in $|e\rangle\langle e|$, which partitions the nodes of $DT(G)$ into computational nodes C, dummy nodes D and trap nodes T.
4: The server sends ρ to the client.
5: The client measures each dummy d in the Z basis, and notes the correction Z^{r_d} for the neighbors, where r_d is the measurement outcome.
6: The client measures the nodes in C to perform the computation while respecting the corrections introduced by the outcomes of the previous measurements and the offsets for the input.
7: The client measures the trap nodes T in the X basis and aborts if there is a node t such that the measurement outcome r_t does not fulfill $r_t = r_{N_{DT(G)}(t)} := \bigoplus_{d\in N_{DT(G)}(t)} r_d$, where $N_{DT(G)}(t)$ denotes the neighborhood of t.
8: If the client does not abort, it corrects the outcome of the computation before outputting it.

Similar to Refs. [17,9,22], we start by showing stand-alone security. In the proof (given in the Appendix), we show an upper bound for the probability of accepting an output that is in a subspace orthogonal to the honest outcomes. Composable security follows directly from the reduction to local criteria as shown in Ref. [8]. However, this reduction creates an overhead and in order to obtain an implementation that is close to the ideal resource, the stand-alone security would need to be high. To achieve this, one could, for example, use a majority vote on the classical outputs of multiple instances of the protocol or fault-tolerant codes such as the RHG code [27] for the computation. A relevant result uses the majority vote for noisy BQP computations [19]. Notably, this protocol does not require enlarging the resource state to include traps for BQP computations; since the input and output are both classical, one can consider pure computation rounds without traps and use the other rounds

for traps. Furthermore, the security proof does not rely on the protocol being in the prepare-and-send setting. Hence, the analysis directly encompasses the receive-and-measure variant in which the client measures in the Z-basis for the dummy nodes in trap rounds as in Protocol 1.

The third type of implementation of $\mathbf{S}^{\mathrm{ver}}$ utilises stabilizer measurements on the resource state. In the RM setting, stabilizer measurements allow the client to verify the state that the server sent [12], leveraging the fact that the resource state is a graph state [13] and hence a stabilizer state. Generators of the stabilizer set of a graph state for a graph $G=(V,E)$ are given by

$$g_j = X_j \otimes \left(\bigotimes_{i \in N_G(j)} Z_i \right) \otimes \left(\bigotimes_{i \in V \setminus (N_G(j) \cup \{j\})} \mathbb{1}_i \right)$$

for all $j \in V$. To translate this mechanism in the PS setting, the server would need to perform measurements on the graph state; however, Z measurements cannot be blindly delegated as they cannot be hidden with angle offsets.

Nevertheless, any stabilizer measurement containing Z on the original graph state is equivalent to the client applying the Z rotation to the sent qubit and then blindly delegating a stabilizer measurement containing only X and Y elements. Let $g=\{\sigma_n\}_{n\in V}$ be a randomly chosen stabilizer of the resource state with $\sigma_n \in \{\mathbb{1},X,Y,Z\}$, i.e.,

$$\left(\bigotimes_{n\in V} \sigma_n \right) \mathcal{E}_G \left(\bigotimes_{n\in V} |+_n\rangle \right) = \mathcal{E}_G \left(\bigotimes_{n\in V} |+_n\rangle \right),$$

where $\mathcal{E}_G$ is a product on CZ gates, one for each edge in G. Now, let $\zeta(n)=1$ if $\sigma_n = Z$ and 0 otherwise. Using commutation of Z and CZ, we find

$$\begin{aligned} &\left(\bigotimes_{n\in V} \sigma_n \right) \left(\bigotimes_{n\in V} Z_n^{\zeta(n)} \right) \left(\bigotimes_{n\in V} Z_n^{\zeta(n)} \right) \mathcal{E}_G \left(\bigotimes_{n\in V} |+_n\rangle \right) \\ &= \left(\bigotimes_{n\in V} \sigma_n^{\zeta(n)\oplus 1} \right) \mathcal{E}_G \left(\bigotimes_{n\in V} Z_n^{\zeta(n)} |+_n\rangle \right) = \mathcal{E}_G \left(\bigotimes_{n\in V} Z_n^{\zeta(n)} |+_n\rangle \right). \end{aligned}$$

Hence, when in the original RM protocol the client tests a stabilizer g containing Z, in the PS setting it can apply a Z gate on the qubits on which Z would otherwise act, and replace the Z stabilizer element with the identity. The resulting stabilizer measurement is now in the XY plane and can be blindly delegated to the server.

3.2 Collective remote state preparation

The third component is collective remote state preparation, which is implemented in the PS setting in Refs. [16,14]. This is a resource that involves n clients and a server. Here we implement a corresponding resource in RM. To do so, we first introduce the ideal resource, which is slightly different than the one in Ref. [14], in that now all clients have some access to the resource [4].

If all clients are honest (denoted with bit $c_j=0$), a client C_k is able to prepare a state vector $|+^\theta\rangle$ at the outer interface of the server, where θ remains secret. If however, some of the clients are dishonest (denoted by $c_j=1$), they can input states ρ_j. One of these states (e.g. the state from the client C_ℓ with the highest identifier) is used for the output so that the state $Z^\theta(\rho_\ell)$ is outputted to the server instead of $|+^\theta\rangle$.

Resource 1 Remote state preparation (RSP)

Input: C_k inputs θ. All other clients C_j input $c_j \in \{0,1\}$ and a qubit register ρ_j.
Output: ρ_S to the server.

1: **if** $\forall j, j\neq k : c_j = 0$ **then**
2: $\quad \rho_S = |+^\theta\rangle\langle+^\theta|$
3: **else**
4: $\quad \ell = \max\{j | c_j = 1\}$
5: $\quad \rho_S = Z^\theta(\rho_\ell)$
6: **end if**

[4] The reason why we needed to introduce a new ideal resource is that the one introduced in Ref. [14] cannot be implemented in the case of dishonest clients and honest server

For each client $j \neq k$, we also consider the filter $\natural_i$ that inputs $c_j = 0$ and the state $|0\rangle\langle 0|$. We consider the following protocol:

Protocol 2 Implementation of RSP in RM.

Input: Client k inputs θ.
Output: The server outputs a qubit register ψ_S.

1: The server prepares the register ψ_S in $|+\rangle\langle +|$ and for each client C_j a register ψ_j in $|0\rangle\langle 0|$.
2: The server applies for each client CX with ψ_S as control and ψ_j as target register.
3: The server sends ψ_j to corresponding client C_j.
4: **for** each client C_j **do**
5: C_j samples $\theta^i \leftarrow_\$ \{k\pi/8 \,|\, 0 \leq k < 8\}$
6: C_j measures the received register in $\{|+^{-\theta_j}\rangle\langle +^{-\theta_j}|, |-^{-\theta_j}\rangle\langle -^{-\theta_j}|\}$, the result is r_j.
7: **if** $j \neq k$ **then**
8: C_j sends (θ_j, r_j) to C_k.
9: **else**
10: C_k receives (θ_j, r_j) from all clients.
11: C_k samples $b \leftarrow_\$ \{0,1\}$ and computes $\delta = (-1)^b\theta - \sum_{i=1}^n \theta_i - \pi \bigoplus_{i=1}^n r_i$.
12: C_k sends b, δ to the server.
13: **end if**
14: **end for**
15: The server receives the correction δ from the client C_k, applies $X^b Z^\delta$ on ψ_S and outputs the register ψ_S.

We find correctness by observing that, for each register ψ_j of the clients, $Z^{\theta_j + r_j\pi}$ is applied on ψ_S. Hence, the final correction that the server applies brings the register in the state $|+^\theta\rangle\langle +^\theta|$. We now assume that the server is honest and some clients are dishonest. We define with $\mathcal{D}$ the set of dishonest and $\mathcal{H}$ the set of honest clients. As the server first entangles the registers, the state the distinguisher gets in the above implementation is:

$$\rho_{\text{impl}} = \left(\left(\bigotimes_{j\in\mathcal{H}} \langle 0_j| + e^{i(\theta_j + r_j\pi)} \langle 1_j| \right) \otimes \left(\bigotimes_{j\in\mathcal{D}} \mathbb{1}_j \right) \otimes X_S^b Z_S^{(-1)^b\theta - \theta'} \right) \left(\bigotimes_{j=1}^n CX_{S,j} \right) \left(\left(\bigotimes_{j\in\mathcal{H}} |0_j\rangle \right) \otimes \left(\bigotimes_{j\in\mathcal{D}} |0_j\rangle \right) \otimes |+_S\rangle \right)$$

where $\theta' = \sum_{j=1}^n \theta_j + r_j\pi$. Using the functionality of the circuit, the fact that Z-rotations on the control register commute with CX and $X^b Z^{(-1)^b\theta} = Z^\theta X^b$ we find:

$$\rho_{\text{impl}} = \left(\left(\bigotimes_{j\in\mathcal{D}} \mathbb{1}_j \right) \otimes Z_S^\theta X_S^b Z_S^{-\theta_\mathcal{D}} \right) \left(\bigotimes_{j\in\mathcal{D}} CX_{S,j} \right) \left(\left(\bigotimes_{j\in\mathcal{D}} |0_j\rangle \right) \otimes |+_S\rangle \right)$$

where $\theta_\mathcal{D} = \sum_{j\in\mathcal{D}} \theta_j + r_j\pi$. The simulator emulates the protocol of the server and the classical part of the protocol of C_k with input $\theta = 0$. It inputs the state it gets from the server to the interface of the dishonest client with the highest identifier. Hence, we need to consider two rounds of corrections, the first is applied by the simulator with $\delta = -\theta_\mathcal{D}$ (since the simulator sets $\theta = 0$), and the second is applied by the ideal resource with Z_S^θ. We then find:

$$\rho_{sim} = \left(\left(\bigotimes_{j\in\mathcal{D}} \mathbb{1}_j \right) \otimes Z_S^\theta \right) \left(\left(\bigotimes_{j\in\mathcal{D}} \mathbb{1}_j \right) \otimes X^b Z_S^{-\theta_\mathcal{D}} \right) \left(\bigotimes_{j\in\mathcal{D}} CX_{S,j} \right) \left(\left(\bigotimes_{j\in\mathcal{D}} |0_j\rangle \right) \otimes |+_S\rangle \right) = \rho_{\text{impl}}$$

If the server is dishonest, we denote the register the simulator receives from the ideal resource as $\psi_I = |+_\theta\rangle\langle +_\theta|$. The simulator implements the protocol of all honest clients but C_k and inputs $c_j = 0$ for all dishonest clients. Instead of implementing the protocol of C_k, the simulator applies $CX_{k,I}$ on the register ψ_k it receives from the server and ψ_I. After that the simulator measures ψ_I in $\{|0\rangle\langle 0|, |1\rangle\langle 1|\}$ and saves the outcome as b. This combination of $CX_{k,I}$ followed by the measurement is equivalent to applying $Z_k^{(-1)^b\theta}$ on ψ_k and is the main mechanism exploited for the PS version of the implementation. At last, the simulator finds r_k by measuring the register ψ_k in $\{\pm_{\theta_k}\}$ with a random θ_k, computes

$$\delta = -\sum_{j=1}^n \theta_j + \pi \bigoplus_{j=1}^n r_j \tag{3}$$

and sends $\delta, b\oplus 1$ to the distinguisher at the server's interface. Note, that for dishonest clients the distinguisher and for honest clients but k the implementation inside the simulator provides θ_j,r_j. We find the following map

$$\frac{1}{2\cdot 8^{|\mathcal{H}|}} \sum_{\substack{\theta\in\mathcal{A}^{|\mathcal{H}|} \\ \mathbf{r}\in\{0,1\}^{|\mathcal{H}|} \\ b\in\{0,1\}}} \mathrm{Tr}\Big\{\Big|+^{-\theta_k-(-1)^b\theta-r_k\pi}\Big\rangle\Big\langle+^{-\theta_k-(-1)^b\theta-r_k\pi}\Big|\psi_k\Big\}\left(\prod_{\substack{j\in\mathcal{H}\\ j\neq k}}\mathrm{Tr}\big\{\big|+^{-\theta_j-r_j\pi}\big\rangle\big\langle+^{-\theta_j-r_j\pi}\big|\psi_j\big\}\right)$$
$$\left|-\sum_{j=1}^{n}\theta_j-\pi\bigoplus_{j=1}^{n}r_j,b\oplus 1\right\rangle\left\langle-\sum_{j=1}^{n}\theta_j-\pi\bigoplus_{j=1}^{n}r_j,b\oplus 1\right|$$

in the simulation. As θ_k and b are uniformly distributed we can first replace $\theta_k\mapsto\theta_k+(-1)^b\theta$ and second $b\mapsto b\oplus 1$ which gives us

$$\frac{1}{2\cdot 8^{|\mathcal{H}|}} \sum_{\substack{\theta\in\mathcal{A}^{|\mathcal{H}|} \\ \mathbf{r}\in\{0,1\}^{|\mathcal{H}|} \\ b\in\{0,1\}}} \left(\prod_{j\in\mathcal{H}}\mathrm{Tr}\big\{\big|+^{-\theta_j-r_j\pi}\big\rangle\big\langle+^{-\theta_j-r_j\pi}\big|\psi_j\big\}\right)\left|(-1)^b\theta-\sum_{j=1}^{n}\theta_j-\pi\bigoplus_{j=1}^{n}r_j,b\right\rangle\left\langle(-1)^b\theta-\sum_{j=1}^{n}\theta_j-\pi\bigoplus_{j=1}^{n}r_j,b\right|$$

which is exactly the map the implementation performs on the registers obtained from the server.

4 Conclussion

Delegated quantum computation is crucial for enabling clients with limited quantum resources to securely and efficiently outsource complex quantum tasks to more powerful quantum servers. Protocols implementing specific functionalities have been proposed in the two prevalent communication settings, *prepare-and-send* (PS) and *receive-and-measure* (RM), and essentially use three components. Amongst these, only the simpler one, namely single-client blind DQC, is known to be perfectly implemented in both settings. Three techniques were used to implement the second component, which is single-client verifiable DQC. The first verification technique, cut-and-choose, i.e., intertwining verifiable computations with the actual computation in multiple rounds of blind DQC, inherits its equivalence from single-client blind DQC. The second technique leverages partitioning the resource state into multiple segments, allowing the use of some segments for verification while others serve for the actual computation. While an implementation of this technique in RM was proposed in 2015 [12], the development of the PS implementations [9,17] imposed a gap between the two settings. We closed this gap by proposing an RM version of [17] which is more efficient than [12]. In contrast to the protocol in [12], our implementation does not require delegating a precomputation to split up the resource state, but utilizes measurements in the Z-basis following the same intuition as the PS implementations that leverage preparation in this basis. Our approach can be easily adapted for other protocols using this verification technique, such as [18]. The last verification technique used for single-client verifiable DQC is stabilizer testing [22]. Although intuitively stabilizer testing requires the client to measure the resource, we translated this technique to PS by leveraging the blindness of the server and deriving an equivalence between stabilizer tests in the two settings. We finally give an RM version of the third component, the remote state preparation protocol from Ref. [16]. This implementation directly implies RM versions of protocols in which this component was used as a subroutine, such as [16] and [14].

We have therefore demonstrated that the two communication models are, in fact, interchangeable in delegated quantum computation; for any (present or future) protocol that is proposed in one setting and consists of these three main components, there exists an equivalent one in the other setting, achieving the same levels of security. Our work not only clarifies the connection between the two communication settings and provides new protocols for DQC, but additionally opens paths for further research. For example, it enables research on previously unexplored hybrid communication settings, where clients that belong to different communication settings (e.g., a preparing client and a receiving client), can collaborate in order to delegate a multiparty computation to a server [16]. Our work also inspires further research of protocols like [10], in which the client sends weak coherent pulses instead of single photons, and raises the question of whether the communication direction in this semi-classical setting is crucial for security. Finally, and also in light of recent implementations and technological advancements, we hope to motivate and simplify the exploration of further protocols to achieve practical delegated quantum computing.

5 Acknowledgments

The authors acknowledge support from the BMBF (QR.X and QR.N), the European Union (Quantum Internet Alliance), the DFG (Emmy Noether grant No. 418294583), the Einstein Research Unit on Quantum Devices and Berlin Quantum.

References

1. Badertscher, C., Cojocaru, A., Colisson, L., Kashefi, E., Leichtle, D., Mantri, A., Wallden, P.: Security limitations of classical-client delegated quantum computing. In: Advances in Cryptology – ASIACRYPT 2020 (2020). https://doi.org/10.1007/978-3-030-64834-3_23
2. Briegel, H.J., Browne, D.E., Dür, W., Raussendorf, R., Van den Nest, M.: Measurement-based quantum computation. Nature Phys. **5**, 19–26 (2009). https://doi.org/10.1038/nphys1157
3. Broadbent, A.: How to verify a quantum computation. Th. Comp. **14**(1), 1–37 (2018). https://doi.org/10.4086/toc.2018.v014a011
4. Broadbent, A., Fitzsimons, J., Kashefi, E.: Universal blind quantum computation. 50th Ann. IEEE Symp. Found. Comp. Sc. **2009**, 517 (2009). https://doi.org/10.1109/focs.2009.36
5. Broadbent, A., Karvonen, M.: Categorical composable cryptography. In: Lecture Notes in Computer Science, pp. 161–183. Springer International Publishing (2022). https://doi.org/10.1007/978-3-030-99253-8_9
6. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: 42nd Ann. IEEE Symp. Found. Comp. Sc. pp. 136–145 (2001). https://doi.org/10.1109/SFCS.2001.959888
7. Colisson, L., Markham, D., Yehia, R.: All graph state verification protocols are composably secure (2024). https://doi.org/10.48550/arXiv.2402.01445
8. Dunjko, V., Fitzsimons, J.F., Portmann, C., Renner, R.: Composable security of delegated quantum computation. In: Advances in Cryptology – ASIACRYPT 2014 (2014). https://doi.org/10.1007/978-3-662-45608-8_22
9. Fitzsimons, J.F., Kashefi, E.: Unconditionally verifiable blind quantum computation. Phys. Rev. A **96**, 012303 (2017). https://doi.org/10.1103/physreva.96.012303
10. Garnier, M., Leichtle, D., Music, L., Ollivier, H.: Composably secure delegated quantum computation with weak coherent pulses (2025), https://arxiv.org/abs/2503.08559
11. Greganti, C., Roehsner, M.C., Barz, S., Morimae, T., Walther, P.: Demonstration of measurement-only blind quantum computing. New J. Phys. **18**, 013020 (2016). https://doi.org/10.1088/1367-2630/18/1/013020
12. Hayashi, M., Morimae, T.: Verifiable measurement-only blind quantum computing with stabilizer testing. Phys. Rev. Lett. **115**, 220502 (2015). https://doi.org/10.1103/PhysRevLett.115.220502
13. Hein, M., Eisert, J., Briegel, H.J.: Multiparty entanglement in graph states. Phys. Rev. A **69**, 062311 (2004). https://doi.org/10.1103/PhysRevA.69.062311
14. Kapourniotis, T., Kashefi, E., Leichtle, D., Music, L., Ollivier, H.: Asymmetric secure multi-party quantum computation with weak clients against dishonest majority. Quant. Sc. Tech. **10**(2), 025015 (2025). https://doi.org/10.1088/2058-9565/adaf12
15. Kashefi, E., Music, L., Wallden, P.: The quantum cut-and-choose technique and quantum two-party computation (2017), https://arxiv.org/abs/1703.03754
16. Kashefi, E., Pappa, A.: Multiparty delegated quantum computing. Crypt. **1**, 12 (2017). https://doi.org/10.3390/cryptography1020012
17. Kashefi, E., Wallden, P.: Optimised resource construction for verifiable quantum computation. J. Phys. A **50**, 145306 (2017). https://doi.org/10.1088/1751-8121/aa5dac
18. Leichtle, D., Music, L., Kashefi, E., Ollivier, H.: Verifying bqp computations on noisy devices with minimal overhead. PRX Quantum **2**(4), 040302 (2021)
19. Leichtle, D., Music, L., Kashefi, E., Ollivier, H.: Verifying bqp computations on noisy devices with minimal overhead. PRX Quantum **2**, 040302 (Oct 2021). https://doi.org/10.1103/PRXQuantum.2.040302
20. Maurer, U.: Constructive Cryptography – A New Paradigm for Security Definitions and Proofs. In: Theory of Security and Applications. pp. 33–56. Springer Berlin Heidelberg, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27375-9_3
21. Maurer, U., Renner, R.: Abstract cryptography. In: Innovations in Computer Science. Tsinghua University Press (2011)
22. Morimae, T.: Verification for measurement-only blind quantum computing. Phys. Rev. A **89**, 060302 (2014). https://doi.org/10.1103/PhysRevA.89.060302
23. Morimae, T.: Measurement-only verifiable blind quantum computing with quantum input verification. Phys. Rev. A **94**, 042301 (Oct 2016). https://doi.org/10.1103/PhysRevA.94.042301
24. Morimae, T., Fujii, K.: Blind quantum computation protocol in which alice only makes measurements. Phys. Rev. A **87**, 050301 (2013). https://doi.org/10.1103/physreva.87.050301
25. Polacchi, B., Leichtle, D., Carvacho, G., Milani, G., Spagnolo, N., Kaplan, M., Kashefi, E., Sciarrino, F.: Experimental verifiable multi-client blind quantum computing on a qline architecture (2024), https://arxiv.org/abs/2407.09310
26. Portmann, C., Renner, R.: Security in quantum cryptography. Rev. Mod. Phys. **94**(2), 025008 (Jun 2022). https://doi.org/10.1103/revmodphys.94.025008
27. Raussendorf, R., Harrington, J., Goyal, K.: A fault-tolerant one-way quantum computer. Ann. Phys. **321**(9), 2242–2270 (2006). https://doi.org/10.1016/j.aop.2006.01.012
28. Raussendorf, R., Briegel, H.J.: A one-way quantum computer. Phys. Rev. Lett. **86**, 5188–5191 (May 2001). https://doi.org/10.1103/PhysRevLett.86.5188
29. Wei, Y.C., Stas, P.J., Suleymanzade, A., Baranes, G., Machado, F., Huan, Y.Q., Knaut, C.M., Ding, S.W., Merz, M., Knall, E.N., Yazlar, U., Sirotin, M., Wang, I.W., Machielse, B., Yelin, S.F., Borregaard, J., Park, H., Lončar, M., Lukin, M.D.: Universal distributed blind quantum computing with solid-state qubits. Science **388**(6746), 509–513 (2025). https://doi.org/10.1126/science.adu6894

A Appendix

Here we examine the stand-alone security of Protocol 1. If the server is dishonest, we can assume that it also has a register S in addition to the received state $|e\rangle\langle e|$. Without loss of generality we further assume that the server first prepares the qubits of the resource state and then applies a joint unitary Ω on all the registers in its possession. After that, it entangles the registers of the resource state and the client's input into the dotted triple-graph, using operation $\mathcal{E}$. The state at the end of the protocol is given by

$$B(\nu)=\mathrm{Tr}_S\left(\sum_s|s\rangle\langle\psi_s^\nu|C_{\nu_C,s}\mathcal{E}\Omega\left(|0\rangle\langle 0|^{\otimes|S|}\otimes\left(\bigotimes_{n\in C,T,D}|+_n\rangle\langle+_n|\right)\otimes|e\rangle\langle e|\right)\Omega^\dagger\mathcal{E}^\dagger C_{\nu_C,s}^\dagger|\psi_s^\nu\rangle\langle s|\right),$$

where s is the vector of measurement results of the client, ν denotes the coloring and the encryption parameters used to obtain $|e\rangle\langle e|$, ψ_s^ν represents the corresponding measurement bases and $C_{\nu_C,s}$ represents the correction on the output that the client applies at the end of the protocol.

Note that, depending on the measurement outcomes of the dummies adjacent to the trap, the client needs to adapt the measurement basis for the traps corresponding to both the input and the resource state. Hence, the client accepts only if the measurement outcome of all traps is 0. We denote with $P_\perp$ the projector into the subspace that is orthogonal to the honest output. We find

$$p_{\text{fail}}=\sum_\nu p(\nu)\mathrm{Tr}\left(\left(P_\perp\otimes\left(\bigotimes_{t\in T}|0_t\rangle\langle 0_t|\right)\right)\mathrm{Tr}_S(B(\nu))\right),$$

where $p(\nu)$ denotes the probability of choosing a specific ν. Performing the trace over the auxiliary system S of the server turns the unitary Ω into a CPTP map, which can be expressed using the Pauli operators σ_i with complex coefficients $\alpha_{k,i}$,

$$p_{\text{fail}}=\sum_{\nu,s,i,j,k}p(\nu)\alpha_{k,i}\alpha_{k,j}^*\cdot$$
$$\mathrm{Tr}\left[\left(P_\perp\otimes\left(\bigotimes_{t\in T}|0_t\rangle\langle 0_t|\right)\right)|s\rangle\langle\psi_s^\nu|C_{\nu_C,s}\mathcal{E}\sigma_i\left(\bigotimes_{n\in C,T,D}|+_n\rangle\langle+_n|\otimes|e\rangle\langle e|\right)\sigma_j\mathcal{E}C_{\nu_C,s}^\dagger|\psi_s^\nu\rangle\langle s|\right].$$

Terms in which σ_i and σ_j are tensor products of $\mathbb{1}$ and X cannot contribute to the sum, except when X acts on the input. We define E to be the subset of Pauli operators, that have at least one Y or Z on any of the registers or X on the input. If the base graph was used in a fault-tolerant setting, the number of operators in $\{Y,Z\}$ (or X on the input) would need to be the number of errors tolerated by the error detection code. However, without fault-tolerance, a single operator can map the output in the orthogonal subspace.

Using cyclicity of the trace, assuming the attack mapped the state into an orthogonal subspace and defining s' as the substring of measurement results of nodes in D and C, we find

$$p_{\text{fail}}\leq\sum_{\nu,s',k}\sum_{i,j\in E}p(\nu)\alpha_{k,i}\alpha_{k,j}^*\mathrm{Tr}\left[\left(\bigotimes_{t\in T}|\psi_t^\nu\rangle\langle\psi_t^\nu|\otimes|\psi_{s'}^\nu\rangle\langle\psi_{s'}^\nu|\right)\mathcal{E}\sigma_i\left(\bigotimes_{n\in C,T,D}|+_n\rangle\langle+_n|\otimes|e\rangle\langle e|\right)\sigma_j^\dagger\mathcal{E}^\dagger\right],$$

where the random flips r_t for all the trap measurements and the random offset θ_t for the input traps are represented by ψ_t^ν. Note that we dropped $C_{\nu_C,s}$ as it only acts on the output nodes which are already projected and traced out.

Following Refs. [17,9], we utilize blindness in the next step; no matter which channel the server applies, the client's registers appear totally mixed to the server, when considering the sum over the outcomes. This implies

$$p_{\text{fail}}\leq\sum_{\nu,k}\sum_{i,j\in E}p(\nu)\alpha_{k,i}\alpha_{k,j}^*\mathrm{Tr}\left[\left(\bigotimes_{t\in T}|\psi_t^\nu\rangle\langle\psi_t^\nu|\right)\sigma_i\left(\bigotimes_{t\in T}|\psi_t^\nu\rangle\langle\psi_t^\nu|\otimes\frac{\mathbb{1}}{\mathrm{Tr}(\mathbb{1})}\right)\sigma_j^\dagger\right].$$

If $\sigma_j\neq\sigma_i$, they either differ on the output registers or on traps. In the first case, the trace vanishes as all Pauli operators except the identity have trace 0. In the second case the trace vanishes, since

$$\sum_{r_t}\sum_{\theta_t}\frac{1}{16}\mathrm{Tr}\big(\langle\psi_t^\nu|\sigma_{i|t}|\psi_t^\nu\rangle\langle\psi_t^\nu|\sigma_{j|t}|\psi_t^\nu\rangle\big)=0,$$

where $\sigma_{i|t}$ and $\sigma_{j|t}$ are single-qubit Pauli operators on trap t. Note that if the trap does not correspond to the input, there is no dependency on θ_t. Hence, only terms with $\sigma_i=\sigma_j$ contribute to the sum, and, therefore, we find

$$p_{\text{fail}}\leq\sum_{k,\nu^T}\sum_{i\in E}p(\nu^T)|\alpha_{k,i}|^2\prod_{t\in T}\sum_{r_t,\theta_t}\frac{1}{16}\big(\langle\psi_t^\nu|\sigma_{i|t}|\psi_t^\nu\rangle\big)^2,$$

where ν has been split up into its contributing terms (i.e., positioning of the traps ν^T, r_t and θ_t). Except for differences in the notation, this is equation (C.10) from [17]. As also the number and types of non-trivial attacks (element in E) are bijectively related, we can refer to the proof in Ref. [17] from here on and find that $p_{\text{fail}}\leq 8/9$.