Lecture Notes

Quantum Cryptography Week 10: Further topics

Contents

10.1 Delegated computation

As powerful as quantum computers may be, in the foreseeable future they are likely to remain rather bulky machines dedicated to special-purpose computations. The first person to have realized the potential of quantum mechanics for computation is Feynman, who thought that one of the main applications of quantum computing devices would be precisely to the problem of *simulation*: quantum computers to simulate quantum systems [**feynman1982simulating**]! Indeed predicting the properties of physical materials such as superconductors, or certain chemical molecules, are some of the most challenging problems faced by experiments, and running accurate simulations of such systems on a classical computer is far too costly.

Today we see Feynman's idea approaching realization. The largest quantum computers to date number in the hundreds of qubits, but these computers can only perform specialized tasks: for instance, they might be arranged in a 2D grid where one is only allowed to apply certain gates on neighboring qubits. Nevertheless, such large bulky machines are in principle universal, meaning that any computation can be translated into one that can be executed on the machine, with a possibly rather large (but nevertheless polynomial) blow-up.

This sets the stage for the scenario of delegated computation. Suppose a user Alice has a quantum circuit $\mathscr{C}$ in mind, that she would like to execute on some input x in order to learn the outcome $\mathscr{C}(x)$. All of Alice's data, x and $\mathscr{C}$, is classical: we can assume x is a classical bit string, and $\mathscr{C}$ is specified by a sequence of two-qubit gates. We can also assume the outcome is classical: let's say it's the result of measuring a specially designated output qubit of $\mathscr{C}$ in the standard basis.

Now, unfortunately Alice herself does not have a universal quantum computer. Maybe she has a tiny desktop machine that for example lets her play around with BB'84-like operations: prepare or measure single qubits, possibly store a couple qubits at a time in memory, but no more. Luckily Alice does have the possibility of buying computation time on a quantum server, with which she could interact over the internet, or maybe even over a simple BB'84-type quantum communication channel that would allow the exchange of one qubit at a time. So Alice could just send x and the description of $\mathscr{C}$ to the server, who would perform the computation, and return the outcome, right?

But remember this is a crypto class. Alice might not trust this server. For one she'd like to have a way to verify that the outcome provided to her is correct. What if the server is lazy and systematically claims the outcome of her computation, $\mathscr{C}(x)$, equals '0'? Since Alice has no quantum computer herself she has no means of checking the outcome! A second property Alice could require is that the computation be private: while she certainly wants to learn $\mathscr{C}(x)$, she'd rather not have the server know that she was interested in circuit $\mathscr{C}$, or in input x, as these might contain private data.

Let's restate these conditions a bit more formally as the requirements that the computation be *correct*, *verifiable* and *blind*.

> **Definition 10.1.1 — Delegated computation.** In the task of delegated computation, a user Alice (sometimes called the *verifier*) has an input $(x, \mathscr{C})$, where x is a classical string and $\mathscr{C}$ the classical description of a quantum circuit. Alice has a multiple-round interaction with a quantum server (sometimes also called *prover*). At the end of the interaction, Alice either returns a classical output y, or she aborts. A protocol for delegated computation is called:
> - *Correct* if whenever both Alice and the server follow the protocol, with high probability Alice accepts (she does not abort) and $y = \mathscr{C}(x)$.
> - *Verifiable* if for any server deviating from the protocol, Alice either aborts or returns $y = \mathscr{C}(x)$.
> - *Blind* if for any server deviating from the protocol, at the end of the protocol the server has no information at all about Alice's input $(x, \mathscr{C})$.

The properties of verifiability and blindness are stated rather informally. A precise defini-

tion satisfying all the desired properties (and in particular, universal composability) would take us many pages. Such a definition was given using the framework of *abstract cryptography* in [**dunjko2014composable**].

The informal definition given above will be sufficient for our purposes. Note that in spite of being rather similar neither of the properties of verifiability or blindness implies the other. In practice it will often be the case that verifiability follows from blindness by arguing using "traps" that, if a protocol is already blind, the server's trustworthiness can be tested by making it run "fake" computations for which Alice already knows the output, without the server being able to distinguish when it is asked to do a real or fake computation. We will see an example of this technique later on.

What are good protocols for delegating quantum computations? It turns out that we don't have a fully satisfactory answer yet: this is an active area of research! Many interesting ideas are being pursued, but none is perfect. In these notes we'll survey the three most prominent approaches. The first construction shows how arbitrary quantum circuits can be delegated, as long as the verifier has the ability to prepare certain specific single-qubit states and communicate them to the server. The second construction achieves a similar result, using a very different idea: it is based on the measurement-based model for quantum computation, which we will introduce. The third construction has a wholly different flavor. It manages to achieve delegation by a purely classical verifier, with no quantum abilities whatsoever. However, the downside is that the verifier now has to interact with *two* isolated quantum servers. This method relies on similar techniques as we have seen in the analysis of device-independent QKD (in particular the use of the CHSH game for testing that the two servers share EPR pairs).

10.1.1 Preliminaries on efficient quantum computation

Before we delegate quantum computations, let's first review briefly what a quantum computation *is*. As hinted at earlier, there are many distinct models for quantum computation, each capable of universal computation (and thus of simulating any other). We'll see three of them in these notes.

The most natural model is called the quantum circuit model. Here a computation is represented by the action of a circuit $\mathscr{C}$ on n input qubits. The input qubits are initialized in an arbitrary quantum state that contains the input of the computation; we will mostly be concerned with classical inputs of the form $|x\rangle$, where $x \in \{0,1\}^n$, but quantum inputs can also be considered. The circuit $\mathscr{C}$ itself is specified by a list of m gates, which are one- or two-qubit unitaries that act on a subset of the qubits. For instance, a simple 4-qubit circuit could be specified as the ordered list $((H,1),(\text{CNOT},(2,3)),(H,3),(Z,4))$. It is then usually assumed that the first qubit is measured in the standard basis to produce a single bit as the classical outcome of the circuit (though quantum outputs can be considered as well). We will write $\mathscr{C}(x)$ for the outcome of a measurement of the output qubit of $\mathscr{C}$ in the standard basis, when $\mathscr{C}$ is executed on the state $|x\rangle$. Thus $\mathscr{C}(x)$ should be treated as a random variable, but we'll assume that for each possible input x it takes on a certain value with high probability, and we'll write $\mathscr{C}(x)$ for that fixed value.

The class BQP of "efficiently computable functions" in the quantum circuit model is formally defined as follows.[1]

> **Definition 10.1.2 — BQP.** A family of functions $\{f_n : \{0,1\}^n \to \{0,1\}\}$ is in BQP if for every integer n there exists a circuit $\mathscr{C}_n$ whose description size (number of gates and precision of the entries in each gate) is polynomial in n and such that for every $x \in \{0,1\}^n$, a measurement of the output qubit of the execution of C_n on $|x\rangle$ returns $f(x)$ with probability at least $2/3$.

The definition of BQP allows arbitrary circuits $\mathscr{C}$, as long as they can be efficiently specified as a list of single- or two-qubit gates. An important theorem in quantum computing, the Solovay-Kitaev

---

[1]In complexity theory we often talk about "languages" rather than functions, were a language is a subset $L \subseteq \{0,1\}^*$. It is then natural to associate a Boolean function to any language, and vice-versa.

theorem, states that it is in fact possible to restrict the set of gates allowed to some simple sets of gates, called "universal gate sets": a gate set is called universal if any circuit that has an efficient implementation, has an efficient implementation that uses only gates from that set. An example gate set that we will use later on is the set

$$\mathscr{G} = \left\{ G = \begin{pmatrix} \cos(\pi/8) & -\sin(\pi/8) \\ \sin(\pi/8) & \cos(\pi/8) \end{pmatrix}, \mathrm{CNOT} \right\},$$

where G implements a $\pi/4$ rotation around the y axis of the Bloch sphere and CNOT a controlled-Z operation (on any two qubits of the circuit). Another example of a popular gate set is the set

$$\mathscr{G}' = \left\{ H, T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}, \mathrm{CNOT} \right\},$$

but there are many others; which gate set to use depends on the task at hand (as well as the architecture of the quantum computer, as certain physical implementations can implement certain gates much more easily than others).

We will use one more useful feature of quantum (as well as classical) circuits, which is the notion of a "universal" circuit. Due to the universality of quantum circuits for efficient quantum computation, and its equivalence with the model of quantum Turing machines, it is possible to show the following.

> **Theorem 10.1.1 — Universal circuit.** For any integer n and size parameter s there exists a fixed circuit $\mathscr{C}_U$ acting on $n + m$ qubits, where m is a polynomially bounded function of n and s, such that for any circuit $\mathscr{C}$ of size at most s expressed using the gate set $\mathscr{G}$, and any input $x \in \{0,1\}^n$ to $\mathscr{C}$, there is a $z \in \{0,1\}^m$ (efficiently computable from $\mathscr{C}$) such that $\mathscr{C}_U(x,z)$ has the same distribution as $\mathscr{C}(x)$.

10.1.2 The Pauli group and Clifford gates

We've already encountered the 4 single-qubit Pauli matrices, $\mathscr{P} = \{I, X, Y, Z\}$. These form a group, in the sense that for any two Pauli matrices P and Q, the product PQ is again a Pauli matrix. A single-qubit *Clifford gate* U is any operation that preserves the Pauli group, in the sense that $UPU^\dagger$ is a Pauli matrix for any Pauli P. If U is a two-qubit gate, we similarly require that $UPU^\dagger \in \pm\mathscr{P}^{\otimes 2}$ for any $P \in \mathscr{P}^{\otimes 2}$.

> **Exercise 10.1.1** Clearly the Pauli matrices are Clifford gates. Show that the Hadamard, phase $P = T^2$ and CNOT gates are both Clifford gates. Do you see other examples? Is the G gate considered above a Clifford gate? How about the T gate? ∎

The defining property of Clifford gates is very useful, and plays an important role in delegated computation. Unfortunately it turns out that there is no universal gate set made only of Clifford gates — any universal set of gates for quantum computation must include at least one non-Clifford gate. As we will soon see this will be a source of many headaches when trying to implement delegated computation.

10.2 Verifiable delegation of quantum circuits

Our first approach to delegating computation is based on the idea of *computing on encrypted data*. Recall the quantum one-time pad from Week 1. Suppose we have a n-qubit density matrix ρ. To encrypt it using the one-time pad we select two n-bit strings $a, b \in \{0,1\}^n$ uniformly at random, and return $\tilde{\rho} = X^a Z^b \rho (X^a Z^b)^\dagger$, where X^a denotes applying a Pauli X operator on all qubits i such that

$a_i = 1$; similarly for Z^b with Pauli Z operators. If ρ represents the input x to the circuit, $\rho = |x\rangle\langle x|$, and the result is classical, $\tilde{\rho} = |x \oplus a\rangle\langle x \oplus a|$. So this is an operation the client Alice can easily perform by herself, and send $\tilde{\rho}$ to the server. If she keeps a local copy of the strings a, b but does not communicate them to the server her input x remains perfectly private.

Now let's see how Alice could make the server execute a circuit $\mathscr{C}$ by acting directly on the encrypted state $\tilde{\rho}$. The goal is to find a transformation $\tilde{\mathscr{C}}$ for the server to apply, such that $\tilde{\mathscr{C}}(\tilde{\rho}) = \widetilde{\mathscr{C}(\rho)}$, an encrypted version of $\mathscr{C}(\rho)$ from which Alice will nevertheless be able to infer the output $\mathscr{C}(x)$. (As we will see, this will also require Alice to update the one-time pad keys she used to encrypt ρ in the first place.)

The circuit $\mathscr{C}$ can be expressed using gates taken from a universal gate set, for example the set $\mathscr{G}' = \{X, H, \text{CNOT}, T\}$ considered in Section 10.1.1. Suppose the first gate in $\mathscr{C}$ is an X to be applied on the second qubit. We'd like the server to evaluate

$$X^a Z^b (X \rho X^\dagger)(X^a Z^b)^\dagger = X^{a \oplus e_2} Z^b \rho (X^{a \oplus e_2} Z^b)^\dagger = X \tilde{\rho} X^\dagger,$$

where e_2 is the bit string with a single 1 in the second position. These equations show that the server doesn't even need to apply the X gate: it is sufficient for Alice to update her one-time pad key from (a, b) to $(a \oplus e_2, b)$. So the X gate is easy. You can see that any Pauli gate, single- or multi-qubit, will be similarly easy.

Let's move one step further and consider the implementation of a Clifford gate — let's take the example of a Hadamard gate on the second qubit, H^{e_2}. Using $HXH = Z$, observe that

$$(X^a Z^b)(H^{e_2} \rho (H^{e_2})^\dagger)(X^a Z^b)^\dagger = (-1)^{a_2 b_2} H^{e_2} X^{a'} Z^{b'} \rho (X^{a'} Z^{b'})^\dagger,$$

where (a', b') is obtained from (a, b) by exchanging the bits a_2 and b_2. Hence if Alice instructs the server to apply an H gate on the second *encrypted* qubit, the effect is the same as if the server had applied the H gate directly on the second *unencrypted* qubit — as long as she updates her one-time pad key (a, b) to (a', b') as described above. The following exercise asks you to show that a similar trick can be employed for any Clifford gate.

> **Exercise 10.2.1** Let U be any one- or two-qubit Clifford gate. Show that the effect of applying U to the encrypted state $\tilde{\rho}$ is equivalent to the application of U on ρ, up to an update rule on the one-time pad key (a, b). Work out the update rule in the case of the phase and CNOT gates. ∎

So Alice can run the whole computation by herself? Remember from Section **??** that no set of Clifford gates is universal — we need to show how to implement one more gate, for instance the T gate considered in the universal set $\mathscr{G}'$. Because the T gate is non-Clifford, applying it to the encrypted state $\tilde{\rho}$ will have a more complicated effect, that we can't keep track of by a simple modification of the one-time pad keys. Instead, we'll show how Alice can make the server implement a T gate on the encrypted state by using the idea of *magic states*.

10.2.1 Computation with magic states

The idea for magic states is that the computation of certain complicated gates on an arbitrary state can be replaced by a simple computation using certain ancilla states called "magic states". Let's see this for the T gate, the only gate that we still need to figure out how to implement. The magic state we need is

$$|\pi/4\rangle = T|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{e^{i\pi/4}}{\sqrt{2}}|1\rangle.$$

Note that preparing this state itself requires applying a T gate. But the point is that we only need to apply the gate to a fixed, known input state, that is independent of the state $|\psi\rangle$ on which we really

want to apply the T gate. So the preparation of single-qubit magic states is a relatively simple task that we could ask Alice to perform by herself.

Suppose we are given a single-qubit state $|\psi\rangle$ on a register A_1, and initialize ancilla register A_2 in the $|\pi/4\rangle$ state. Consider the following circuit: first, apply a CNOT, controlled on A_2 and acting on A_1. Second, measure register A_1 in the computational basis, obtaining an outcome c. Third, apply a gate P^c to register A_2. What is the corresponding post-measurement state of register A_2? It is a good exercise to verify that this is $X^c Z^c T |\psi\rangle$. That is, up to a simple (XZ) correction, we applied the T gate to $|\psi\rangle$ while only requiring a magic state, a CNOT, and a measurement in the computational basis.

(Thomas: include picture of T gate gadget)

> **Exercise 10.2.2** Suppose that instead of being applied directly to the state $|\psi\rangle$, the circuit described above is applied to an encrypted version of $|\psi\rangle$, $X^a Z^b |\psi\rangle$. Show that the outcome of the circuit is then $P^a X^{a'} Z^{a'} T |\psi\rangle$, for some bits a' and b' depending on a, b and c. (Convince yourself that the same calculation works out in case $|\psi\rangle$ is not pure, but a reduced density ρ on a single qubit. ∎

Note the phase gate that we picked up in the exercise. This also needs to be corrected. But the phase is applied on the encrypted state. Alice could instruct the server to apply P^a, but this would require revealing the bit a, which is part of the key. There is a way around this that involves adding a little bit of randomization in the choice of magic state we use, so as to guarantee that the phase correction to be applied is always independent of the one-time pad key. You can try to work this out as an exercise, or look at the paper [] for a possible solution.

10.2.2 Blindness

With the main ingredients in place, there are still a number of difficulties we need to overcome. Let's first consider the blindness property. For this, we assume the server is completely honest, and we're only worried that it may be able to obtain information about Alice's input x or circuit $\mathscr{C}$, while still following the correct protocol.

Unfortunately it seems clear that the circuit itself has to be revealed to the server: in the scheme we described in the previous section each application of a T gate will involve an interaction between the client and the server. But there is a simple way out: Alice can instruct the server to execute a fixed "universal" circuit (as in Theorem 10.1.1), and instead encode the actual circuit she is interested in as part of the input. (Note that this method is not very practical, as there is a large computational overhead in using such a universal circuit.)

So now the only thing that Alice needs to keep hidden is her new input (x, z), and this is precisely what the one-time pad achieves. But we should be careful: even though the OTP certainly hides the input at the start of the interaction, subsequent interactions could in principle reveal more information. The only gate which involves an interaction between Alice and the server is the T gate, which requires Alice to tell the server whether a P correction should be applied or not. As we mentioned earlier it is possible to choose a magic state uniformly at random from a set of 8 possibilities, in a way that both the magic state and the correction bit will appear uniformly random to the server, irrespective of the measurement outcome it reports to Alice. Once again we refer you to [] for the (simple) details on how this can be performed.

10.2.3 Verifiability

The protocol we described satisfies the blindness property (provided we use a universal circuit and implement the T gates as sketched above), but so far it is not verifiable: the client has no guarantee that the server performs the required computation; indeed, no check is performed and the

client would accept any answer (note that under the OTP, any string can be interpreted as a valid ciphertext).

The idea to implement verifiability is to combine the protocol with some "test runs". The original protocol is now called a "computation run". In a test run the client knows what the outcome should be, and she will check that the server returns the correct value. But the server will not be able to distinguish test runs from computation runs, and as a consequence we'll have the guarantee that the server is also being honest in a computation run.

There are two types of test runs, X-test and Z-test. In an X-test run, the computation is executed on an encryption of the all-0 input $|0\rangle^{\otimes n}$. In a Z-test run, the same computation is executed on an encryption of $|+\rangle^{\otimes n}$. The main trick to ensure that the verifier is able to keep track of the computation is that all gates in a test run are replaced by *identity* gates, without the prover making the difference! Note that we already know how to do this for Pauli gates, as these did not involve the server anyways. The H gate requires a bit more work, but the idea is simple: since an H exchanges the standard basis and the Hadamard basis we can simply think of it as exchanging between an X-test run to an Z-test run, so that the verifier can still perfectly keep track of the state that the circuit should be in. The T gate, of course, is the interesting one. The idea is to modify the implementation described in Section 10.2.1 by changing the magic state, as well as the update rule, and in a way that is un-noticeable by the server but will result in an application of the identity gate instead of the T. The following exercise asks yu to work out how this can be done.

> **Exercise 10.2.3** Consider the following procedure for implementing a T gate on the single-qubit state $|\psi\rangle_{A_1}$, using a magic state. The client first selects two bits $d, y \in \{0,1\}$ uniformly at random, and prepares the magic state $Z^d P^y T|+\rangle_{A_2}$, where $P = T^2$ is the phase gate. The client sends system A_2 to the server, who performs a CNOT, controlled on A_1 and with target A_2. The server measures A_1 in the computational basis, obtaining an outcome $c \in \{0,1\}$ that it sends to the client. The client then sends back $x = y \oplus c$ to the server, who applies a gate P^x to the remaining system A_2.
>
> 1. Show that the state of A_2 at the end of this procedure is $X^c Z^{c(y \oplus 1) \oplus d \oplus y} T|\psi\rangle$, i.e. it is an encryption (using a key known to the client) of $T|\psi\rangle$.
>
> Next let's suppose we're doing a computation run, so that $|\psi\rangle = X^a|0\rangle$ for some $a \in \{0,1\}$. The client would like to perform the identity instead of a T gate, without the server noticing. This can be done by executing precisely the same circuit, except the magic state is replaced by $X^d|0\rangle$ (it does not depend on y).
>
> 2. Show that with the magic state replaced by $X^d|0\rangle$ the interaction results in a register A_2 in state $X^d|0\rangle$. Show that in this case the outcome c of the server's measurement is deterministically related to a and d in a simple way.
>
> 3. Can you find a similar modification, with a different magic state, that will implement the identity for the case of a Z-test run, where $|\psi\rangle_{A_1} = Z^b|+\rangle$ for some $b \in \{0,1\}$? ∎

The exercise shows that simply by changing the magic state used in the implementation of the T gate, the client can force that gate to act as identity in an X- or Z-test run. Moreover, due to the random bits d, y used in the preparation of the magic state you can verify that, from the point of view of the server, these magic states look uniformly distributed, and thus it has no way of telling which "gadget" — for a T gate or the identity — it is really implementing.

In a test run the verifier knows exactly what the outcome of the circuit should be, so that it can verify the answer provided by the client. Is this enough to ensure that the server cannot cheat in a computation run? After all, we can imagine that the server may be able to perform certain attacks that do not affect simple computations, where the state is always a tensor product of single qubits encoded in the computational or Hadamard bases, but such that the attack would perturb the kind

of highly entangled states that will show up at intermediate stages in a more complex circuit.

To show that this is not the case — that any significant attack will necessarily have a noticeable effect on either the X- or Z-test runs, the idea is to use an observation called the "Pauli twirl", that you are asked to work out in the next exercise.

> **Exercise 10.2.4 — Pauli twirl.** Let ρ be a single-qubit density matrix, and $P, P' \in \mathscr{P}$, where $\mathscr{P} = \{I, X, Y, Z\}$ is the set of single-qubit Pauli operators. Show that $\sum_{Q \in \mathscr{P}}(Q^\dagger P Q)\rho(Q^\dagger (P')^\dagger Q)$ equals $P\rho P^\dagger$ if $P = P'$, and is 0 otherwise. Show that the same result holds for n-qubit Pauli operators. ∎

The Pauli twirl lets us argue that, thanks to the use of the quantum one-time pad, any "attack" of the server boils down to the application of a Pauli operator at the last step of the circuit. Indeed, suppose first that the interaction performed between the client and the server results in the correct circuit $\mathscr{C}$ being implemented, except at the last step the server applies an arbitrary "deviating unitary" U. Thus the outcome is $U\tilde{C}\tilde{\rho}\tilde{C}^\dagger U^\dagger$, where $\tilde{C}$ is the unitary Alice instructed the server to implement, and $\tilde{\rho}$ the initial OTP-encoded state sent by the client. Due to the OTP, $\tilde{\rho}$ has the form $\tilde{\rho} = \sum_{Q \in \mathscr{P}} Q|x\rangle\langle x|Q^\dagger$, where $|x\rangle$ denotes the real input state that the client would like the computation to be performed on. Moreover, for any Q there is a correction $c(Q) \in \mathscr{P}$ applied by the client, which is such that $c(Q)\tilde{C}Q|x\rangle\langle x|Q^\dagger\tilde{C}^\dagger(c(Q))^\dagger = C|x\rangle\langle x|C^\dagger$. Thus, after applying $c(Q)$ to the corrupted circuit,

$$\sum_{Q \in \mathscr{P}} c(Q)U\tilde{C}Q|x\rangle\langle x|Q^\dagger\tilde{C}^\dagger U^\dagger(c(Q))^\dagger = \sum_{Q \in \mathscr{P}} c(Q)U(c(Q))^\dagger c(Q)\tilde{C}Q|x\rangle\langle x|Q^\dagger\tilde{C}^\dagger(c(Q))^\dagger c(Q)U^\dagger(c(Q))^\dagger$$

$$= \sum_{Q \in \mathscr{P}} c(Q)U(c(Q))^\dagger C|x\rangle\langle x|C^\dagger c(Q)U^\dagger(c(Q))^\dagger$$

$$= \sum_{P \in \mathscr{P}} |\alpha_P|^2 PC|x\rangle\langle x|C^\dagger P^\dagger,$$

where for the last step we decomposed $U = \sum_{P \in \mathscr{P}} \alpha_P P$ in the Pauli basis, and used the property of the Pauli twirl proved in Exercise 10.2.4.

This computation shows that any unitary applied by a malicious server at the end of the honest circuit is equivalent to a convex combination of Pauli operators applied at the end of the circuit. But any such non-trivial operator will be detected in either the X- or Z-test runs, as it will result in one of the outcomes being flipped in either the standard or Hadamard bases.

To conclude we need to deal with the case where the server applies a deviating unitary, not at the end of the circuit, but at some intermediate step. But this case can be reduced to the former! Indeed, we can always think of a "purified" version of the whole protocol, where all measurements are deferred until the end. Suppose the unitary $\tilde{C}$ the server is supposed to implement decomposes as $\tilde{C} = \tilde{C}_2\tilde{C}_1$, and the server applies a deviating unitary U in-between the two circuits. The result can be written as

$$\tilde{C}_2 U \tilde{C}_1 = (\tilde{C}_2 U \tilde{C}_2^\dagger)\tilde{C}_2\tilde{C}_1,$$

where we used that $\tilde{C}_2$ is unitary, $\tilde{C}_2^\dagger\tilde{C}_2 = \mathbb{I}$. Thus the deviation U is equivalent to applying another deviating unitary $U' = \tilde{C}_2 U \tilde{C}_2^\dagger$ at the end of the circuit, and we are back to the analysis performed in the previous case: if the deviation has a non-trivial effect it will be detected by the client in one of the test runs.

10.3 Delegation in the measurement-based model

Our second scheme for delegated computation has a similar flavor to the one presented in the previous section, but at its heart it is based on a completely different approach to universal quantum computation. So far we have encountered the circuit model for performing quantum computations.

From the point of view of computer science this is the most natural model, as it is a direct analogue of the classical circuit model on which the architecture of our (classical) computers is based. However, quantum information allows for other, much more exotic, models of computation. Many of these models were originally proposed with the idea that they might be more powerful than the circuit model, although ultimately they were proved equivalent. This includes the adiabatic model for computation [**aharonov2008adiabatic**] and the measurement-based model that we will now study.

The highlight of measurement-based quantum computation (MBQC) is that it allows one to implement an arbitrary quantum computation (specified by a circuit using some universal gate set) solely by executing an (adaptive) sequence of single-qubit measurements on a fixed, universal starting state. Seems impossible? Let's first give an overview of how this model works, and then we'll explain how it can be used to achieve blind, verifiable delegated computation.

10.3.1 Measurement-based computation

Measurement-based computation is based on an idea very similar to *teleportation-based computation*, a model to which we return in the next section. It is the idea that a complete quantum computation, including the preparation of the initial state and the application of gates from a universal set, can be performed by making a sequence of adaptive measurements on a fixed universal state, simultaneously "teleporting" the input state from one qubit to the next while at the same time applying unitary transformations on the state.

Let's do a simple example first. Suppose we have a qubit initialized in the state $|\psi\rangle_A = \alpha|0\rangle + \beta|1\rangle$. Suppose a second qubit is created in the state $|+\rangle_B$, and a CTL-Z operation is performed, controlling on the first qubit to perform a phase flip on the second. Then the joint state of the system is $|\psi\rangle_{AB} = \alpha|0\rangle|+\rangle + \beta|1\rangle|-\rangle$. Suppose now we measure the first qubit in the Hadamard basis. What happens to the second qubit? Let's re-write

$$
\begin{aligned}
|\psi\rangle_{AB} &= \alpha|0\rangle|+\rangle + \beta|1\rangle|-\rangle \\
&= \frac{1}{\sqrt{2}}|+\rangle(\alpha|+\rangle + \beta|-\rangle) + \frac{1}{\sqrt{2}}|-\rangle(\alpha|+\rangle - \beta|-\rangle).
\end{aligned}
$$

The measurement rule states that if we get the outcome "+" the second qubit is projected to $|\psi\rangle_B = \alpha|+\rangle + \beta|-\rangle$, and if we get a "−" it is projected to $\alpha|+\rangle - \beta|-\rangle$. In the first case, $|\psi\rangle_B = H|\psi\rangle_A$, and in the second $|\psi\rangle_B = XH|\psi\rangle_A$. More succinctly put, $|\psi\rangle_B = X^m H|\psi\rangle_A$ where $m \in \{0, 1\}$ denotes the outcome of the measurement, $m = 0$ in case of "+" and $m = 1$ in case of "−". Thus, up to a "Pauli correction" X^m, we managed to apply a Hadamard gate simply by making a single-qubit measurement on the appropriate state.

For an arbitrary $\phi \in [0, \pi/2]$ let

$$
|+_\phi\rangle = \frac{1}{\sqrt{2}}|0\rangle + e^{i\phi}\frac{1}{\sqrt{2}}|1\rangle \qquad \text{and} \qquad |-_\phi\rangle = \frac{1}{\sqrt{2}}|0\rangle - e^{i\phi}\frac{1}{\sqrt{2}}|1\rangle, \tag{10.1}
$$

and

$$
U_z(\phi) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}, \qquad U_x(\phi) = HU_z(\phi)H.
$$

The rotations $U_z(\phi)$ and $U_x(\phi)$ together generate a universal set of single-qubit gates, as any rotation on the Bloch sphere can be implemented as $U_z(\varphi_3)U_x(\varphi_2)U_z(\varphi_1)$ for an appropriate choice of φ_1, φ_2 and φ_3. The following exercise asks you to generalize our example of the Hadamard gate to any single-qubit rotation that can be decomposed in this way.

> **Exercise 10.3.1** Modify the method we described to apply a Hadamard gate by instead performing a measurement of the first qubit in the basis $\{|+_\varphi\rangle, |-_\varphi\rangle\}$. Show that the second qubit is then projected on the state $X^m H U_z(\varphi)|\psi\rangle_A$, where $m \in \{0,1\}$ indicates the measurement outcome.
>
> Now consider a sequence of three measurements with angles φ_1, φ_2 and φ_3. That is, suppose a first qubit is in state $|\psi\rangle_A$, and three additional qubits are created in the $|+\rangle$ state and organized on a line. Three CTL-Z operations are performed from left to right. Then the first qubit is measured in basis $\{|+_{\varphi_1}\rangle, |-_{\varphi_1}\rangle\}$, obtaining an outcome $m_1 \in \{0,1\}$, the second qubit is measured with angle φ_2, obtaining outcome m_2, and finally the third qubit is measured with angle φ_3, obtaining outcome m_3. Show that the state of the fourth qubit can then be written as
>
> $$|\psi\rangle_D = X^{m_3} Z^{m_2} X^{m_1} H U_z((-1)^{m_2}\varphi_3) U_x((-1)^{m_1}\varphi_2) U_z(\varphi_1)|\psi\rangle_A. \tag{10.2}$$
>
> [Hint: you may use the identities $XU_z(\phi) = U_z(-\phi)X$ and $HU_z(\phi)H = U_x(\phi)$, valid for any real ϕ.] ∎

The exercise *almost* lets us apply an arbitrary rotation $U_z(\varphi_3)U_x(\varphi_2)U_z(\varphi_1)$, except there are these annoying "corrections", the X and Z operations and the Hadamard to the left, and the extra $(-1)^{m_i}$ phases in the angles. But these can be dealt with easily! For the phases, note that we perform the measurements sequentially, and the phase flip that got applied to a certain angle only depends on the outcome of the measurement performed right before. For the case of the calculation performed in the exercise, if we *really* had wanted to end up with $U_x(\varphi_2)$, after having obtained outcome m_1 we would have updated our choice of angle in which to measure to $\varphi_2' = (-1)^{m_1}\varphi_2$. As for the X, Z and H corrections at the end of the computation, we can handle those at the time of final measurement: they correspond to corrections that will need to be applied once we measure the final qubit (this is similar to how we handled the one-time pad in the previous section).

This shows how any sequence of single-qubit rotations can be applied to a qubit. You would start with a line of m qubits, each initialized in the $|+\rangle$ state. Then apply CTL-Z operations on all pairs of neighboring qubits, from left to right. This corresponds to preparing a $1 \times m$-dimensional "brickwork state", our universal resource for single-qubit computation. Suppose for simplicity the initial qubit is meant to be initialized in the $|+\rangle$ state (if it is not you can modify the circuit so that the first gate applied prepares the correct qubit). Any rotation can be applied by decomposing it in the form $U_z(\varphi_3)U_x(\varphi_2)U_z(\varphi_1)$ and making the correct sequence of measurements on three qubits, keeping track of successive measurement outcomes to update the angles and the X, Z and H "corrections" that tag along to the left of the description of the state of the qubit, as in (10.2) (note that you do not need to remember all measurement outcomes, but only their combined effect in terms of a power of X and a power of Z).

What if we have a multi-qubit computation? We won't give the details, as the overall idea is the same. Since we already know how to implement arbitrary single-qubit gates, to get a universal gate set it suffices to implement a 2-qubit CNOT gate. We'll use multiple lines of qubits, one per qubit of the computation. The lines are connected by vertical CTL-Z operations once every three qubits (in a slightly shifted manner). A two-qubit CNOT gate can then be applied using similar ideas as we described, but performing measurements on the two lines associated with both qubits on which the gate acts. We'll leave the details as an exercise, and refer you to the notes [**browne2006one**] for detailed explanations.

10.3.2 Blind delegation in the MBQC model

Now that we have seen how to perform an arbitrary computation in the MBQC model, let's see how the computation can be delegated to an untrustworthy server. Let's imagine that the client, Alice,

has a sequence of single-qubit measurements, specified by angles $\varphi_{i,j}$ and update rules (depending on prior measurement outcomes), that she wishes to apply on an $n \times m$ brickwork state in order to implement a certain circuit she is interested in. Let's also assume for simplicity that the outcome of the last measurement would (possibly after a Pauli correction if needed) give her the answer she is looking for.

Of course Alice could tell the server to prepare the $n \times m$ brickwork state and then instruct it, through a classical interaction, to perform the measurements specified by the $\varphi_{i,j}$. The server would report the outcomes, Alice would perform the updates, and tell the server the next angle to measure in. But clearly this would be neither blind nor verifiable.

The key idea is then for Alice to (partially) prepare some kind of "one-time padded" version of the brickwork state, on which the server will implement the computation without ever having any information about the "real" angles $\varphi_{i,j}$.

Consider the following outline for a protocol.

Protocol 1 Fix a set of "hiding" angles $D = \{0, \pi/4, 2\pi/4, \dots, 7\pi/4\}$.
1. For each of the nm qubits of the brickwork state, Alice chooses a random $\theta_{i,j} \in D$, prepares the state $|+_{\theta_{i,j}}\rangle$, and sends it to the server.
2. The server arranges all qubits it receives in the shape of an $n \times m$ brickwork state, and performs CTL-Z operations on neighboring qubits as required.
3. Alice and the server have a classical interaction over nm rounds. In each round,
    (a) Alice computes an angle δ_{ij} as a function of θ_{ij}, φ_{ij}, private randomness r_{ij}, and previous outcomes b_{ij} reported by the server. She sends δ_{ij} to the server.
    (b) The server measures the (i, j)-th qubit of the brickwork state in the $\{|+\rangle_{\varphi_{ij}}, |-\rangle_{\varphi_{ij}}\}$ basis and reports the outcome $b_{ij} \in \{0, 1\}$ to the client.
4. Alice infers the outcome of her circuit from her private data and the server's last reported outcome.

There are many details missing to fully specify the protocol. The idea is to design rules for Alice to update the measurement angles δ_{ij} that she sends to the server in a way that, from the point of view of the server δ_{ij} is uniformly random in D (so it reveals no information about the computation being performed), but Alice is able to keep track, under her one-time pad, of the actual computation. To see how this can be done, first consider the following exercise.

Exercise 10.3.2 Based on Exercise 10.3.1 we know that applying a Hadamard gate to a qubit A can be performed by measuring the qubit in the basis $\{|+\rangle, |-\rangle\}$, and adding an X^m correction, where m is the measurement outcome.

This is correct when the second qubit, B has been initialized in a $|+\rangle$ state, as required for the un-hidden brickwork state. Now suppose the qubit has in fact been initialized in the state $|+_\theta\rangle$, for some real angle θ (and a CTL-Z operation has been performed on the two qubits). Show that the result of measuring the first qubit in the basis $\{|+_\delta\rangle, |-_\delta\rangle\}$ is to project the second qubit on the state $X^m H U_z(\theta + \delta)|\psi\rangle_A$, where m is the measurement outcome. ∎

Suppose then that Alice would like to apply a gate $U_z(\varphi)$, for some angle $\varphi \in A$. The exercise shows that by communicating the angle $\delta = \varphi - \theta$ to the server instead, where θ is the initial angle using which she prepared the corresponding qubit of the brickwork state, will have the desired effect of implementing $X^m H U_z(\varphi)$. However, this still poses a problem: if the server is given both the quantum state $|+_\theta\rangle$, *and* the real angle $\varphi - \theta$, we can't argue any more that the computation is blind, as the joint distribution of these two pieces of information depends on φ.

Exercise 10.3.3 Fix φ, and suppose an adversary is given a classical value $\eta = \varphi - \theta$ and a single-qubit state $|\psi\rangle = |+_\theta\rangle$, where θ is chosen uniformly at random. Design a strategy for

the adversary to recover φ, given $(\eta, |\psi\rangle)$. What is its success probability (averaged over the random choice of θ)? ∎

The role of the additional values r_{ij} specified in the protocol is to render φ_{ij} completely hidden to the server. Here r_{ij} is chosen uniformly at random in $\{0, 1\}$, and Alice communicates the angle $\varphi - \theta + r\pi$ to the server. Based on exercise 10.3.2 the effect of $r\pi$ on the computation to add an extra Z^r correction, which Alice can easily keep track of. To see that it is sufficient to ensure blindness, imagine that instead $r\pi$ had been added to the initial angle θ. For any fixed θ, a random choice of $r \in \{0, 1\}$ is sufficient to ensure the server gains no information from receiving $|+_{\theta+r\pi}\rangle$, as $\frac{1}{2}|+_\theta\rangle\langle+_\theta| + \frac{1}{2}|+_{\theta+\pi}\rangle\langle+_{\theta+\pi}| = \frac{1}{2}\mathbb{I}$. But then as θ varies in D the angle $\theta - \varphi$ itself is uniformly distributed in D. Therefore from the point of view of the client the joint distribution of the pair $(|+_\theta + r\pi\rangle, \theta - \varphi)$ is indistinguishable from that of a uniformly random qubit and a uniformly random value from D. The server receives completely random data, so the computation is perfectly blind.

10.3.3 Verifiability

In the previous section we showed how blind delegation could be implemented in the MBQC model. Can we make the protocol verifiable? Note that so far the client does not perform any checks, so the server could just as easily report random outcomes to the client at each step. Already though, due to blindness there is no way the server can *force* a particular outcome on the client; the best it can do is mislead her into thinking that the outcome of the computation is some random bit.

There are different techniques available to make the protocol verifiable, and we refer to [**fitzsimons2012uncondition** for details. The main idea is to introduce *trap qubits*. Those are particular rows of the brickwork state that the client randomly inserts into its circuit but carry no operation: they are meant to remain in the $|0\rangle$ state (hidden, as usual, under the quantum one-time pad). By asking the server to measure a qubit on such a line the client can verify the measurement outcome. Due to the blindness property, even the application of identity gates cannot be detected by the server, thus it does not know it is being tested.

Implementing this idea requires a little care, as it is important to ensure that even the tiniest attack by the server, such as reporting a single false measurement outcome, is detected with good probability, as just one such deviation could suffice to ruin the whole computation. This can be achieved by introducing additional ideas from fault-tolerant computation that we will not go into here.

10.4 Delegating to two servers

Both schemes for delegated computation we've seen so far, in the circuit model or using measurement-based computation, require the client to prepare single-qubit states taken from a small fixed set and send them to the server. What if the client has no quantum abilities whatsoever? Intuitively, the aim of the qubits sent by the client in the two previous schemes is to establish some kind of "trusted space" within the server's quantum memory, in which the computation is to be performed. The quantum one-time pad is used to guarantee that if the server does not use this space then the client interprets the results, at best, as garbage. In fact the verifiability property, enforced through the use of trap qubits, ensures that the server's cheating will be detected with high probability.

How can we establish a "trusted computation space" within the server's memory, without sending it the qubits in the first place? You know the answer! In previous weeks we've seen that simple tests based on the CHSH game could be used to guarantee that *two* arbitrary but non-communicating devices share a specific state, the EPR pair $|\phi^+\rangle$. Even if it is limited to a single qubit per server, this gives us a solid starting point: a test which ensures that a certain little corner of the servers' workspace behaves in a way that we can control.

Let's see how this idea can be leveraged to devise a scheme for delegated computation in which the verifier is completely classical, but has access to *two* non-communicating servers, both untrusted. This method is the most technical of the three we are presenting, and we'll remain at an intuitive level of presentation. If you are interested to learn more we refer you to the main paper on the topic [**reichardt2013classical**].

10.4.1 Establishing a trusted computation space

In Week 7 we saw the CHSH rigidity theorem, which states that if the servers successfully play the CHSH game then up to local isometries the operations they perform are equivalent to those specified in the ideal strategy for the CHSH game. Thus the CHSH game provides a simple test, not only to certify the presence of an EPR pair between the servers, but also the structure of the measurements that the servers perform on their respective half of the EPR pair when asked certain questions. The central idea for using this in delegated computation will be to sometimes play the CHSH game with the servers, and sometimes other games, some of which will involve the computation Alice wants them to implement; this will be done in a way that the servers invidiually can never tell whether they are being "CHSH-tested" or actually being "used" to implement a useful part of the computation, so that they have to apply the honest CHSH strategy all the time, test or computation.

The first thing to deal with is that we're going to need many more EPR pairs. One idea to certify n EPR pairs would be to play n CHSH games "in parallel": Alice could select n pairs of questions $(x_j, y_j)_{j=1,\dots,n}$ to send to the servers, collect n pairs of answers (a_j, b_j), and check how many satisfy the CHSH condition $a_j \oplus b_j = x_j \wedge y_j$. If this estimate is close enough to the optimal $\cos^2 \pi/8$ she would accept the interaction. Although this is a sensible idea it is currently not known how well it works; in particular the effect of small errors in the servers' answers is not clear. (The difficulty is similar to one we encountered in Week 4, when we saw an example of a game for which the servers could play two repetitions of the game much better than you'd expect by using a correlated strategy across both instances of the game.)

Instead of executing the games in parallel Alice will perform them sequentially. That is, she sends the questions (x_j, y_j) to the servers one pair at a time, waiting for their answer before sending the next pair of questions. After having repeated this procedure for n rounds, she counts the average number of rounds in which the CHSH condition was satisfied, and accepts if and only if it is at least $\cos^2 \pi/8 - \delta$, for some error threshold δ. The following sequential rigidity theorem states the consequences of this test in the idealized setting where $\delta = 0$.

> **Theorem 10.4.1 — idealized.** Suppose the two servers, Bob and Charlie, successfully play n sequential CHSH games. Then up to local isometries their initial state is equivalent to $|\phi^+\rangle_{BC}^{\otimes n} \otimes |junk\rangle_{BC}$. Moreover, at each step $j \in \{1,\dots,n\}$ the measurements performed by each server are equivalent to those of the ideal strategy for CHSH (Z and X for Bob and H and $\tilde{H}$ for Charlie) applied on the j-th EPR pair.

You may notice that the protocol for the n sequential CHSH tests is similar to how the CHSH tests are performed in the protocol for device-independent quantum key distribution we saw in Week 7. The analysis uses similar tools: a first step uses a (Martingale) concentration inequality to argue that, if a fraction about $\cos^2 \pi/8 - \delta$ of the games are won by the servers, then for most $j \in \{1,\dots,n\}$ the *a priori* probability that the servers would have won in round j must be of the same order, say at least $\cos^2 \pi/8 - 2\delta$. For any such j the basic CHSH rigidity theorem can be applied to conclude that the measurements applied, and the state on which they were applied, are (up to local isometries) equivalent to the ideal CHSH strategy.

By itself this reasoning is not sufficient to imply that the servers' initial state is a tensor product of EPR pairs. Indeed, the different EPR pairs used in each round could partially "overlap", or

even be the same pair! Intuitively we know this is not possible, as any measurement destroys the EPR pair, so it cannot be re-used. But this is tricky to establish rigorously; the way the errors add up through a proof by induction can be hard to control. Nevertheless, it can be done, and for the remainder of the section we will assume that a "robust" version of the "idealized" theorem above can be proven dealing with the more realistic setting where the servers are not required to play the CHSH games strictly optimally, a far too stringent requirement for any practical application.

10.4.2 State tomography

Now that we have a way to establish a "secure computation space", as a second step let's see how the client Alice can use that space, and additional CHSH tests, to certify that one of the servers has prepared certain single- or two-qubit states in that space.

Consider the following protocol. With Bob, Alice behaves exactly as if she was executing the n sequential CHSH games described in the previous section. With Charlie, however, she does something different: she instructs him to measure each half of the EPR pairs he is supposed to share with Bob in a certain basis, say $\{|+_\theta\rangle, |-_\theta\rangle\}$ for some real theta (see (10.1)), and to report the outcome.

Charlie of course knows that something special is going on. So we have no guarantee as to what action he performs. In contrast, Bob is told the exact same thing as in the n-sequential CHSH test. He must thus behave exactly as if this is the test Alice was performing, and Theorem 10.4.1 applies: in each round, Bob applies the ideal CHSH measurements, in the standard or Hadamard bases, on his half of the j-th EPR pair, in a way that, if Charlie had been measuring using his own CHSH measurements, they would have succeeded with near-optimal probability.

But now Charlie is doing something different — we don't know what. But *if* Charlie performs the measurement asked by Alice, and reports the right outcome, we know what should happen: Charlie's half-EPR pair gets projected onto one of the basis states, $|+_\theta\rangle$ or $|-_\theta\rangle$, and by the special properties of EPR pairs so does Bob's half. In particular, whenever Bob performs a measurement in the Hadamard basis the average value of his outcome (considered as a value in $\{\pm 1\}$) should be $\langle +_\theta | X | +_\theta \rangle = \cos\theta$ or $\langle -_\theta | X | -_\theta \rangle = -\cos\theta$. Thus by collecting all Bob's answers associated to measurements in the X basis Alice can check whether the average outcome over the rounds in which Charlie reported a $+$ is approximately $\cos\theta$, and $-\cos\theta$ over those rounds when Charlie reported a $-$. Alice is using Bob's answers to perform tomography on the state that Charlie claims having prepared, without Bob being able to detect what is going on! (Even though Bob knows he *might not* be currently tested, since he is aware of the structure of the protocol, there is nothing he can do about it — if he deviates he risks failing too many CHSH games, in case this is what Alice is doing.)

In the CHSH game the only measurements made by Bob are in the computational or Hadamard bases. To perform tomography of arbitrary multi-qubit states we would also need him to sometimes apply a Pauli Y. It is possible to do this via a simple modification of the CHSH game. For our purposes the modification will not be necessary, as the set of states that are characterized by their expectation value with respect to Pauli X and Z observables (call such states *XZ-determined*) is sufficient to implement the delegated computation protocol.

> **Exercise 10.4.1** Show that the family of all single-qubit states in the xz-plane of the Bloch sphere, i.e. all states of the form
>
> $$\rho = \frac{1}{2}(I + \cos\theta X + \sin\theta Z), \qquad \theta \in [0, 2\pi),$$
>
> are *XZ*-determined.

Show that the family of two-qubit states of the form

$$|\psi\rangle = U \otimes P|\phi^+\rangle,$$

for any single-qubit real unitary U and $P \in \{I, X, Y, Z\}$, is XZ-determined.

Give an example of two distinct single-qubit states that have the same expectation values with respect to both X and Z observables, and are thus not XZ-determined. ∎

10.4.3 Process tomography

Beyond state tomography, our protocol for delegated computation will require us to implement some limited form of *process tomography*: we need to find a way to guarantee that at least one of the servers, Bob or Charlie, is performing the right computation! At first this task may appear overwhelming: while as described in the previous section it is possible to use one server to perform tomography against the other server's state, how can we test for a certain *gate* being applied? For the case of state preparation we know what the right states are, and as long as they are restricted to simple single- or two-qubit states we can do full state tomography. But our ultimate goal is to implement an arbitrary quantum circuit, which may generate highly entangled states of its n qubits; there is no hope to perform full tomography on such states, as it would require an exponential number of measurements.

We will sidestep the difficulty and use a model of computation which only requires the application of a very special type of gate — a measurement in the Bell basis, i.e. the simultaneous eigenbasis of $X \otimes X$ and $Z \otimes Z$, given by

$$|\psi_{00}\rangle_{AB} = \frac{1}{\sqrt{2}}(|00\rangle_{AB} + |11\rangle_{AB}), \qquad |\psi_{01}\rangle_{AB} = \frac{1}{\sqrt{2}}(|00\rangle_{AB} - |11\rangle_{AB}),$$

$$|\psi_{10}\rangle_{AB} = \frac{1}{\sqrt{2}}(|01\rangle_{AB} + |10\rangle_{AB}), \qquad |\psi_{11}\rangle_{AB} = \frac{1}{\sqrt{2}}(|01\rangle_{AB} - |10\rangle_{AB}),$$

where $|\psi_{00}\rangle = |\phi^+\rangle$ is the familiar EPR pair. This model of computation is called *teleportation-based computation* (recall that a measurement in the Bell basis is precisely the operation required of the sender in the teleportation protocol), and we'll review it in the next section. But let's already see how this idea can be used for delegated computation.

In a similar vein as in the previous section, suppose Alice instructs Charlie to measure his n qubits in the Bell basis, where the qubits are paired in an arbitrary way chosen by Alice (so she tells Charlie the whole set of measurements to be performed at the outset). Of course as usual Charlie does what he wants — he may not even have n qubits in the first place. But Alice also instructs Bob to play sequential CHSH games, so that from his point of view the protocol is perfectly indistinguishable from the tests. Once Alice has collected all of Bob and Charlie's outcomes, she groups Bob's outcomes when they are associated to the same state, and uses them to check that Charlie did not lie. For instance, if Charlie reports $|\psi_{00}\rangle$ then whenever Bob measured the two corresponding qubits using the same basis, computational or Hadamard, his two outcomes should be the same. (Note that not all Bob's measurements are useful, as it will sometimes be the case that the qubits were measured in different bases, in which case there is no useful test Alice can perform — she simply discards those rounds.)

The following exercise asks you to make this argument more formal.

Exercise 10.4.2 Suppose Bob and Charlie share two EPR pairs, $|\phi^+\rangle_{B_1 C_1} \otimes |\phi^+\rangle_{B_2 C_2}$. Charlie measures his two halves, $C_1 C_2$, using an arbitrary four-outcome POVM, obtaining a result $(c_1, c_2) \in \{0, 1\}^2$. Bob measures each of B_1 and B_2 using observables $O_1, O_2 \in \{X, Z\}$ chosen uniformly at random.

Suppose that if $(O_1, O_2) = (X, X)$ then Bob's outcomes (as values in $\{\pm 1\}$) satisfy $b_1 b_2 = a$, and if $(O_1, O_2) = (Z, Z)$ they satisfy $b_1 b_2 = d$, for some fixed values $a, d \in \{\pm 1\}$ (i.e. imagine the same experiment is repeated many times, and Bob's outcomes consistently satisfy these equations). Show that Charlie must have been implementing a measurement in the Bell basis. Which Bell state is associated to each of the four possible values for (a, d)? ∎

The exercise shows that, provided we can trust that Bob and Charlie indeed share EPR pairs, and Bob's measurements are made in the computational or Hadamard bases, then Alice has a way to verify that Charlie has been implementing a Bell basis measurement on certain pre-specified pairs of qubits. Just as for the case of state tomography, these assumptions are guaranteed by the fact that Bob cannot tell the difference between when Alice is executing the process tomography protocol described here, or when she is executing sequential CHSH games.

10.4.4 Teleportation-based computation

The final ingredient needed for our delegation protocol is a method of computation adapted to the kinds of operations we are able to certify of the servers: preparation of EPR pairs and single- or two-qubit XZ-determined states (Exercise 10.4.1), and measurements of pairs of qubits in the Bell basis (Exercise 10.4.2).

Computation by teleportation is a model of computation which allows just that. The main idea is that a gate can be applied to a qubit by "teleporting the qubit into the gate". The following exercise fleshes out the main gadget used in computation by teleportation.

Exercise 10.4.3 Let $|\psi\rangle_A$ be an arbitrary single-qubit state and let $|\phi\rangle_{BC} = (I \otimes UP|\phi^+\rangle)$, where U is an arbitrary single-qubit unitary and $P \in \{I, X, Y, Z\}$. Suppose a measurement of qubits A and B is performed in the Bell basis, yielding a pair of outcomes $(b_1, b_2) \in \{0, 1\}^2$. Show that there exists a Pauli operator Q (depending only on (b_1, b_2)) such that the post-measurement state of the qubit in C is $(UQPU^\dagger)U|\psi\rangle$. ∎

The idea is then the following. Suppose that Alice wishes to implement an arbitrary computation on n qubits, specified by a circuit $\mathscr{C}$ using the universal gate set $\mathscr{G} = \{\text{CNOT}, G\}$ introduced in Section 10.1.1. Assume for simplicity the input to the circuit is $|0\rangle^{\otimes n}$. Alice initializes her work space with a large number of "magic states" from the set

$$\left\{ |0\rangle, (I \otimes H)|\phi^+\rangle, (I \otimes G)|\phi^+\rangle, \text{CNOT}_{B_1 B_2}(|\phi^+\rangle_{A_1 B_1}|\phi^+\rangle_{A_2 B_2}) \right\}. \tag{10.3}$$

At each stage of the computation Alice keeps track of a special set of n qubits which represent the current state of the circuit. We can label these as $A_1 \cdots A_n$, even though they will change over time. Initially $A_1 \cdots A_n$ point to any n of the "magic" $|0\rangle$ qubits she has prepared in her workspace.

Now suppose Alice would like to apply a gate to one of her qubits A_j, for example a G gate. Then she can perform the circuit described in Exercise 10.4.3, where the role of A is played by A_j, and the roles of B and C by one of her "magic" $(I \otimes G)|\phi^+\rangle$. As a result the state of C is projected to $(GQG^\dagger)G|\psi\rangle_C$, where initially A_j is in state $|\psi\rangle$ (the same computation would work for mixed states as well). This is the operation Alice wanted to perform, except for the correction $GQG^\dagger$. How do we deal with this?

Depending on Q, $GQG^\dagger$ will amount to a Pauli correction, possibly multiplied by a Hadamard: $GXG^\dagger = iHY$, $GYG^\dagger = Y$ and $GZG^\dagger = H$. By now we are used to Pauli corrections: Alice can keep track of these as a form of one-time pad that is tagged along the whole computation. The Hadamard gate is a little more annoying, but in fact it can be easily corrected using one more step of "teleportation", this time using a "magic" $(I \otimes H)|\phi^+\rangle$. This will induce yet another correction $HQ'H^\dagger$, but this time whatever Q' is the result is a simple Pauli correction that Alice can tag along as part of the one-time pad.

Thus, aside from the preparation of the magic states, the whole computation boils down to a simple sequence of Bell basis measurements. Note however that, due to the necessity of performing Hadamard corrections in an unpredictable way (as it depends on measurement outcomes obtained when teleporting into a G gate), this sequence is adaptive. This is similar to the scenario of MBQC, but it will require us to proceed with a little extra care in our delegation protocol.

10.4.5 Blind and verifiable delegated computation

With all the ingredients in place we are ready to describe a delegation protocol for the classical client Alice to delegate the computation of an n-qubit quantum circuit $\mathscr{C}$, specified using the universal gate set $\{\text{CNOT}, G\}$, to two isolated quantum servers, Bob and Charlie, sharing entanglement.

> **Protocol 2** Alice compiles a description of $\mathscr{C}$ in the teleportation-based model. Let $T = \text{poly}(|\mathscr{C}|)$ be chosen large enough, and $\delta = \text{poly}^{-1}(|\mathscr{C}|)$ small enough. Alice performs one of the following four tests (the first three with equal probability, and the last with smaller probability depending on the size of $\mathscr{C}$).
>
> 1. She plays T sequential CHSH games with the two servers. If the average number of successes is below $\cos^2(\pi/8) - \delta$ she rejects.
> 2. Alice executes a state tomography protocol in which Charlie is asked to perform measurements that correspond to $T/9$ copies of each of the four magic states in (10.3). Bob is instructed to play T CHSH games. Bob's measurement outcomes are used to check the results reported by Charlie, as described in Section 10.4.2.
> 3. Alice executes a process tomography protocol in which Bob is instructed to perform Bell basis measurements according to the pattern that corresponds to the teleportation-based circuit associated with $\mathscr{C}$. Charlie is instructed to play T CHSH games. Charlie's measurement outcomes are used to verify that Bob is reporting the correct outcomes for his Bell measurements, as described in Section 10.4.3.
> 4. Alice orchestrates the computation of $\mathscr{C}$: she instructs Charlie to prepare the magic states as in step 2., and Bob to perform Bell basis measurements as in step 3. She uses the measurement outcome reported by Bob for the output qubit as the output of the computation (after having applied any required Pauli corrections).

The main reason this protocol works is that Bob cannot distinguish a computation run from a process tomography run, or a CHSH run from a state tomography run; similarly Charlie cannot distinguish between a computation run and a state tomography run, or between a CHSH run and a process tomography run. The protocol can be made blind: blind to Bob, whom without knowledge of which magic states his Bell basis measurements are performed on gains no useful information from the pattern of Bell measurements Alice instructs him to perform (the pattern can be made independent from the circuit $\mathscr{C}$, aside from its size); blind to Charlie, who prepares magic states in a way that is completely independent from the computation. Verifiability follows directly from the tests performed in cases 1., 2. and 3. of the protocol.

There is one difficulty we hinted at earlier and we have glossed over so far. This is the fact that, after application of a Bell basis measurement corresponding to teleportation into a G gate, Alice needs to make an adaptive choice: either apply an H correction, or not. However, Bob should be ignorant of this choice, as otherwise the protocol would no longer be blind. The solution is to switch the focus over to Charlie. While Bob will always be asked to perform the same pattern of Bell basis measurements, Charlie will be (adaptively) asked by Alice to create certain magic states as $|\phi^+\rangle$, and others as $(I \otimes H)|\phi^+\rangle$, as a function of the outcomes reported by Bob. Since these outcomes are uniformly distributed, the pattern of state preparation requests Charlie sees is still random, so that he does not gain any information about the computation either. (Note however that a third observer able to eavesdrop on both the messages exchanged with Bob and with Charlie

would learn valuable information about the computation; however such an attack falls outside of the scope of the security definition of delegated computation.)

Only one task remains: performing a soundness analysis of the protocol! Given it is not possible to require that the servers *exactly* pass all the tests, some error should be tolerated. How does this error effect the quality and trustworthiness of the computation? This is quite delicate. The best analysis known to-date makes this protocol, compared to the ones we saw in the previous two sections, highly inefficient, as it requires T to be a very large power of n before even relatively weak security guarantees can be obtained. Nevertheless, it is the only protocol known for purely classical delegated computation, and improving it is an important research problem.

Acknowledgments