I. Bell and Boris' Bogus Journey

From the pale green of burnt tellurium to the conductivity of copper, quantum mechanics seems necessary to explain the world. But quantum mechanics is also famously weird. As Richard Feynman said,

> *If you think you understand quantum mechanics, you don't understand quantum mechanics.*

Sometimes, a physicist is so squeamish they reject it altogether. Albert Einstein, one of the founders of quantum theory but later its most famous critic, stated

> *God does not play dice.*

In their classic 1935 paper, Einstein, Podolsky and Rosen (EPR) found a way to make the world look quantum-mechanical without a dice-playing God. According to EPR, God runs a clockwork universe, but conceals its workings from the poor schmucks at the other end of oscilloscope. Since these clockwork theories replace fundamental randomness with hidden classical variables obeying a locality condition, they are called *local hidden variable* theories.

At face value, it seems impossible to distinguish quantum mechanics from its clockwork twin. But like twins, they are easier to tell apart when they stand side by side! In 1964, John Bell found a brilliant way to check if Nature runs on clockwork or dice by directly comparing their predictions. The goal of this tutorial will be to explain the math behind Bell's result, and then perform the corresponding experiment on a quantum computer. We can compare twins, and discover if nature is classical or quantum!

Bounding with Bell

Bell's method involves nothing more sophisticated than flipping coins. For simplicity, we will focus on the two-coin example due to Clauser, Horne, Shimony, and Holt (CHSH). Imagine that Alice and Bob are two grad students, hired by Eve to perform the menial chore of flipping coins. Instead of just observing whether the coins are tails or heads, Alice and Bob have *two* different measurements they can choose from. Since we are trying to "mock up" quantum mechanics, we will use quantum-mechanical notation for states and operators, and explain how to restrict to local hidden variable theories below.

Label Alice's two available measurements by A_0, A_1, Bob's measurements by B_0, B_1, and assume that all measurements have outcomes ± 1. For instance, A_0 might measure if the flipped coin is heads $(+1)$ or tails (-1), while A_1 measures if the tail points in a more northerly $(+1)$ or southerly (-1) direction. Although Eve can fiddle with the coins beforehand, once the experiment starts, Alice and Bob cannot influence each others measurements. This means Alice and Bob's measurements *commute*, which we can write succinctly as

$$[A_i, B_j] = A_i B_j - B_j A_i = 0. \tag{1}$$

Since any operator has outcomes ± 1, the outcome of applying any operator twice is the identity:

$$A_i^2 = B_j^2 = \mathbb{I}. \tag{2}$$

You can check this assertion below.

Exercise 1. Show that if a Hermitian observable A has eigenvalues ± 1 (with any multiplicity), then $A^2 = \mathbb{I}$. *Hint.* Consider the eigenbasis of A.

For each run of the experiment, Alice can choose to measure A_0 or A_1, while Bob can choose to measure B_0 or B_1. For many trials, we define the *correlation*

$$\langle A_i B_j \rangle := \lim_{N_{ij}\to\infty} \frac{n_{ij}^+ - n_{ij}^-}{N_{ij}},$$

where $n_{ij}^{\pm}$ is the number of times $A_i B_j = \pm 1$, and $N_{ij} := n_{ij}^+ + n_{ij}^-$ is the total number of trials where Alice chooses A_i and Bob chooses B_j.

Exercise 2. Show that, if Eve prepares a pure state $|\psi\rangle$ for the two-coin system, the correlation is given by

$$\langle A_i B_j \rangle = \langle \psi | A_i B_j | \psi \rangle. \tag{3}$$

Similarly, if Eve prepares a density ρ, the correlation is

$$\langle A_i B_j \rangle = \mathrm{Tr}[\rho A_i B_j].$$

Bell realized that quantum mechanics and classical hidden variable theories make *different* predictions about the correlations between Alice and Bob. To see how, define a new operator involving both Alice and Bob's possible measurements:

$$\mathcal{E} = A_0 B_0 + A_0 B_1 + A_1 B_0 - A_1 B_1.$$

With a little algebra, we can find a useful expression for $\mathcal{E}^2$.

Exercise 3. Using $A_i^2 = B_j^2 = \mathbb{I}$, show that squaring the operator $\mathcal{E}$ gives

$$\mathcal{E}^2 = 4\mathbb{I} - [A_0, A_1][B_0, B_1]. \tag{4}$$

We have, perhaps suspiciously, been using quantum-mechanical formalism to describe classical physics. A simple observation is that *classical measurements commute.* In classical physics—for instance, a local hidden variable theory—observation has no effect on the system, so Alice can perform measurements on her system in any order she pleases, and the same goes for Bob. Thus, $[A_0, A_1] = [B_0, B_1] = 0$. From (4), we have

$$\mathcal{E}^2 = 4\mathbb{I} - [A_0, A_1][B_0, B_1] = 4\mathbb{I}.$$

Taking expectations immediately gives the *CHSH bound*:

$$|\langle \mathcal{E} \rangle| \leq \sqrt{\langle \mathcal{E}^2 \rangle} = 2. \tag{5}$$

This is a bound on classical correlations between Alice and Bob. A *general Bell inequality* is any such bound on classical correlations.

Exercise 4. To derive (5), we used the fact that for any Hermitian operator A,

$$|\langle A\rangle| \leq \sqrt{\langle A^2\rangle}.$$

Prove this is true. *Hint.* Use the fact that $\langle (A - \langle A\rangle)^2\rangle \geq 0$.

One of the nice features of this approach is that, by discussing operators, we have sidestepped classical states altogether. We can even figure out how to max out the bound using operators alone. In fact, I'll leave it to you!

Exercise 5. Check that any *deterministic* assignment of outcomes to A_i, B_j saturates the CHSH bound.

Alice and Bob don't even need to flip their coins! When they do, they introduce *classical randomness* into the experiment. In fact, Eve can even permit Alice and Bob to *share* randomness beforehand, e.g. they flip a third coin and use its outcome to make decisions once the experiment starts.

If Eve only allows Alice and Bob to share classical randomness, their density matrix ρ will be *separable*: there is some set of probabilities $p_k \geq 0$, $\sum_k p_k = 1$, such that

$$\rho = \sum_k p_k \rho_A^k \otimes \rho_B^k.$$

Let's break this down. This distribution $\{p_k\}$ is precisely the shared classical randomness Alice and Bob access before the experiment. If the shared random outcome is k, Alice chooses density matrix ρ_A^k, and Bob chooses ρ_B^k, and then perform their separate measurements. If these separate but jointly conditioned measurements are deterministic, then Alice and Bob will again saturate CHSH, as you can show momentarily. In fact, this completely characterizes the set of experiments which achieve the classical bound!

Exercise 6. Show that, if ρ_A^k and ρ_B^k correspond to deterministic assignments, then Alice and Bob will achieve the CHSH bound.

You might wonder whether classical states and operators are equivalent notions. Our earlier derivation of the CHSH bound said nothing about states, and in fact, even with an entangled state (discuss further below) classical operators still obey CHSH. You need quantum measurements to exploit quantum states. But if we assume a classical state, trying to maximize $|\langle \mathcal{E}\rangle|$ inevitably leads us back to classical operators, since quantum measurements tend to do worse, as we show now.

Exercise 7. Consider a density matrix $\rho = \rho_A \otimes \rho_B$.

(a) Show that

$$\langle \mathcal{E}\rangle = \langle A_0\rangle\langle B_0\rangle + \langle A_0\rangle\langle B_1\rangle + \langle A_1\rangle\langle B_0\rangle - \langle A_1\rangle\langle B_1\rangle.$$

(b) Define the function

$$f(x, y, a, b) := xa + xb + ya - yb.$$

Compute ∇f. Conclude that the only local extremum is $x = y = a = b = 0$.

(c) Writing $x = \langle A_0 \rangle, y = \langle A_1 \rangle, a = \langle B_0 \rangle, b = \langle B_1 \rangle$, argue that to maximize $|\langle \mathcal{E} \rangle|$, we must choose $x, y, a, b \in \{\pm 1\}$.

(d) From part (c), conclude that for separable states, $|\langle \mathcal{E} \rangle| \leq 2$, and moreover the maximising operators are classical.

(e) Extend your results to the general separable case, $\rho = \sum_k p_k \rho_A^k \otimes \rho_B^k$.

When Alice and Bob only share classical randomness, quantum measurements on their individual coins will not help beat the CHSH bound. The best they can do is make classical measurements! Thus, not only do quantum measurements need quantum states, but the converse is true, and quantum states are need to exploit quantum measurements.

We have said that a classical universe runs on clockwork, while the quantum universe is random. So where does classical randomness fit into the picture? Really, classical randomness is due to ignorance rather than fundamental physics. When you roll (classical) dice, the outcome is completely deterministic, though sensitive to initial conditions we do not know or have control over. To paraphrase Einstein, in a classical world, humans can play dice but an omniscient God cannot. In quantum mechanics, randomness is a fundamental feature of the universe, and even God is obliged to join in the game!

Tangling with Tsirelson

Bell's inequality provides a simple way to check if Nature is classical: hire a team of grad students to perform quantum measurements on a bunch of operators to see if (5) is obeyed. Actually, even better than that, you can do this using a real-life quantum computer. But before we do, it's worth asking if are there any bounds on *quantum correlations*. Otherwise, our experiment might not only prove that classical physics is wrong, but quantum mechanics as well! In 1980, Tsirelson found such a constraint. First, we need a few technical results.

Exercise 8. The *operator norm* for a matrix A is just the largest eigenvalue of A, or equivalently,

$$||A|| = \sup_{|\psi\rangle} |A|\psi\rangle|.$$

(a) From the definition, argue that $||AB|| \leq ||A|| \cdot ||B||$.

(b) Show that $\langle A \rangle_\psi = \langle \psi | A | \psi \rangle \leq ||A||$ for any state $|\psi\rangle$. *Hint.* Use Cauchy-Schwarz.

(c) Extend (b) to any density matrix ρ, i.e. $\mathrm{Tr}[\rho A] \leq ||A||$.

(d) Check that $||A||$ satisfies the *triangle inequality*, $||A + B|| \leq ||A|| + ||B||$.

Let's apply these results to (4):

$$\begin{aligned}\langle \mathcal{E}^2 \rangle \leq ||\mathcal{E}^2|| &= ||4\mathbf{I} - [A_0, A_1][B_0, B_1]|| \\ &\leq 4 + ||[A_0, A_1]|| \cdot ||[B_0, B_1]|| \\ &\leq 4 + 4||A_0|| \cdot ||A_1|| \cdot ||B_0|| \cdot ||B_1|| = 8.\end{aligned}$$

This leads to *Tsirelson's bound* on the expectation $\langle \mathcal{E} \rangle$:

$$|\langle \mathcal{E} \rangle| \leq \sqrt{\langle \mathcal{E}^2 \rangle} = 2\sqrt{2}. \tag{7}$$

Anything exceeding Tsirelson's bound is called *super-quantum*. Concretely, you might wonder how to violate Bell's inequality, and saturate Tsirelson's bound. We can kill two bounds with one state!

Exercise 9. Suppose Eve prepares the *singlet* state on the two coins,

$$|\Psi\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle),$$

while Alice and Bob choose the following operators:

$$A_0 = -\frac{Z_A + X_A}{\sqrt{2}}, \quad A_1 = \frac{Z_A - X_A}{\sqrt{2}}, \quad B_0 = X_B, \quad B_1 = Z_B.$$

Show that the resulting correlations saturate Tsirelson's bound, and hence violate Bell's inequality.

Clearly, there is something special about the singlet state $|\Psi\rangle$. In the previous section, we showed that classical states were separable. Since we break the CHSH bound, $|\Psi\rangle$ must not be separable. You can directly confirm this in the next exercise.

Exercise 10. Show that $|\Psi\rangle$ is not separable, i.e. there is no choice of $\{p_k, \rho_A^k, \rho_B^k\}$ for which

$$|\Psi\rangle\langle\Psi| = \sum_k p_k \rho_A^k \otimes \rho_B^k.$$

The opposite of separability is *entanglement*. Since entanglement (non-separability) is needed to violate Bell's inequalities, entanglement is at the heart of quantum correlations. Ironically, although EPR advocated for local hidden variable theories, by introducing entanglement to the world they provided the very resources needed to show the world is truly quantum! Put another way, even when Einstein was wrong, he was right.

References

1. On the EPR Paradox (1964), John Bell.
2. Proposed Experiment to Test Local Hidden-Variable Theories (1969), John F. Clauser, Michael A. Horne, Abner Shimony, and Richard A. Holt.
3. Can Quantum-Mechanical Description of Physical Reality Be Considered Complete? (1935), Albert Einsten, Boris Podolsky and Nathan Rosen.
4. Quantum Generalizations of Bell's inequality (1980), Boris Tsirelson.

II. From Quaternions to Quantum Cryptography

Basic group theory

A symmetry is *change without difference*: a "boring" transformation under which an object looks the same. A *group* is a doodad for keeping track of symmetries. The formal properties of groups are axiomatized as follows:

> ***Definition 1 (groups).*** A set G with binary operation $\cdot$ is a *group* if:
>
> 1. (*Closure*) For all $a, b \in G$, $a \cdot b \in G$.
> 2. (*Associativity*) For every $a, b, c \in G$, $a \cdot (b \cdot c) = (a \cdot b) \cdot c$.
> 3. (*Identity*) There is an *identity* $e \in G$ such that $e \cdot a = a \cdot e = a$, for all $a \in G$.
> 4. (*Inverse*) For each element $a \in G$, there is an *inverse* element $b \in G$ such that $a \cdot b = b \cdot a = e$.

In other words, doing one symmetry, then another, is still boring (closure); symmetries are transformations, and compose as such (associative); there is a trivial symmetry which does nothing (identity); and any symmetry can be done "backwards" (inverse).

Usually, the first example of a group we encounter is the set of integers $G = \mathbb{Z}$ with respect to addition. Soon afterwards, we encounter the set of nonzero reals with respect to multiplication, $G = \mathbb{R}^*$. In both cases, the order in the binary operator doesn't matter, with $a \cdot b = b \cdot a$, where $\cdot$ is the group operation ($+$ for the integers and $\times$ for the reals). Such a group is called *commutative* or *Abelian.* In quantum mechanics, we use matrices everywhere, and since matrix multiplications is not usually commutative, we obtain *non-Abelian* groups.

There are a few more basic definitions we need. The first is the notion of one group being contained inside another:

> ***Definition 2 (subgroups).*** Let $(G, \cdot)$ be a group. A set $H \subseteq G$ is a *subgroup* if it is closed under $\cdot$, the group operation.

We also want to know when two groups have the "same" structure:

> ***Definition 3 (homomorphism and isomorphism).*** Let G, H be groups. A map $\varphi : G \to H$ is a *homomorphism* if
>
> $$\varphi(a \cdot_G b) = \varphi(a) \cdot_H \varphi(b).$$
>
> An *isomorphism* is a bijective homomorphism. We view G and H as "the same" just in case there is an isomorphism between them.

Let's explore these definitions in a familiar example.

Exercise 1. We'll consider the set of unitary $N \times N$ matrices $\mathrm{U}(N)$, defined by $UU^\dagger = U^\dagger U = \mathbb{I}_N$, where $\mathbb{I}_N$ is the $N \times N$ identity.

(a) Show that $\mathrm{U}(N)$ forms a group.

(b) Demonstrate explicitly that for $N > 1$, this group is non-Abelian.

(c) Verify that the map $\varphi : \mathrm{U}(1) \to \mathrm{U}(N)$ given by

$$\varphi(e^{i\theta}) = \begin{bmatrix} e^{i\theta} & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}$$

is a homomorphism.

(d) For each $k < N$, find a subgroup of $\mathrm{U}(N)$ isomorphic to $\mathrm{U}(k)$.

The Pauli group and quaternions

In quantum mechanics, our groups are formed by matrices acting as transformations on Hilbert space. For instance, for n qubits, the Hilbert space has $N = 2^n$ dimensions, and the associated group of unitary matrices (or gates) is $\mathrm{U}(2^n)$. But this is a very large set, and as humans trying to build a quantum computer, it's interesting to see what we can do with simple subgroups of $\mathrm{U}(2^n)$.

We start with the basic example of the *Pauli group*, which as the name suggests, is a generalization of the set of single-qubit Pauli operators X, Y, Z. To make it easier to talk about, let's first introduce a slick notation for groups.

Definition 4 (generators and relations). Often, we can express a group in terms of the products a few basic elements called *generators* $g_1, \ldots, g_n$. The remaining structure of the group can be encoded in *relations* $r_1, \ldots, r_m$ satisfied by the generators. We write the corresponding group

$$G = \langle g_1, \ldots, g_n | r_1, \ldots, r_m \rangle.$$

As physicists, we reserve the right to be sloppy and write only generators, leaving relations implicit.

As a warm-up, we'll describe the Pauli group on a single qubit.

Exercise 2. The Pauli group on a single qubit is more or less identical to the (unit) *quaternions*, a weird number system defined in the 19th century. This exercise explores both.

(a) In 1843, the great Irish mathematician William Rowan Hamilton was walking along the Royal Canal in Dublin, when he was struck by a sudden inspiration. With his penknife, he carved this gnomic inscription on Brougham Bridge:

$$i^2 = j^2 = k^2 = ijk = -1.$$

The objects i, j, k are called *quaternions*, and generalize the imaginary unit $i = \sqrt{-1}$. Show that $Q_8 = \{\pm 1, \pm i, \pm j, \pm k\} = \langle i, j, k\rangle$ forms a group under multiplication, using Hamilton's Brougham Bridge rules.

(b) The (single-qubit) *Pauli group* is generated by Pauli matrices under multiplication:

$$\mathcal{P}_1 = \langle X, Y, Z\rangle.$$

How big is the group?

(c) The Pauli algebra obeyed by the operators is $X^2 = Y^2 = Z^2 = -iXYZ = \mathbb{I}$, where $\mathbb{I}$ is the 2×2 identity matrix. This looks a lot like the Brougham Bridge rules! Find a subgroup of $\mathcal{P}_1$ isomorphic to Q_8.

The *Pauli group on n qubits*, $\mathcal{P}_n$, consists of all tensor products of Pauli operators acting on individual qubits. We use subscripts to indicate which qubit these act on, e.g. if $n = 3$, then $X_1 = X \otimes \mathbb{I} \otimes \mathbb{I}$. So, being maximally sloppy with notation, we have

$$\mathcal{P}_n = \langle X_1, \ldots, X_n, Y_1, \ldots Y_n, Z_1, \ldots Z_n\rangle = \langle X_i, Y_i, Z_i\rangle.$$

The relations are just the Pauli algebra on each qubit, and the multiplication rule is given by $\bigotimes_i A_i \cdot \bigotimes_i B_i = \bigotimes_i (A_i \cdot B_i)$.

Exercise 3. Let's get acquainted with the Pauli group $\mathcal{P}_n$.

(a) Let $\alpha \in \{0, 1\}^n$ be an n-bit string, with $\alpha(i)$ the ith bit. Define the elements

$$X^{(\alpha)} = \bigotimes_{i=1}^{n} X_i^{\alpha(i)}, \quad Z^{(\alpha)} = \bigotimes_{i=1}^{n} Z_i^{\alpha(i)},$$

with $X_i^0 = Z_i^0 = \mathbb{I}$. Derive the relations

$$\begin{aligned} X^{(\alpha)} Z^{(\beta)} &= (-1)^{\alpha\cdot\beta} Z^{(\beta)} X^{(\alpha)} \\ X^{(\alpha)} X^{(\beta)} &= X^{(\alpha\oplus\beta)} \\ Z^{(\alpha)} Z^{(\beta)} &= Z^{(\alpha\oplus\beta)} \end{aligned}$$

where $\alpha \cdot \beta$ is the dot product modulo 2, and $\alpha \oplus \beta$ is bitwise xor as usual.

(b) Show that all elements in $\mathcal{P}_n$ can be written in the form $CX^{(\alpha)}Z^{(\beta)}$ for some $\alpha, \beta \in \{0, 1\}^n$ and $C \in \{\pm 1, \pm i\}$.

(c) Let $\hat{\mathcal{P}}_n = \{X^{(\alpha)}Z^{(\beta)} : \alpha, \beta \in \{0, 1\}^n\}$. What subgroup of $\mathcal{P}_n$ do these elements generate?

Sharing quantum secrets

The Pauli group has many applications. One of the most exciting is to *quantum cryptography*, where we can use the Pauli group to build an unbreakable quantum code! The basic problem is as follows. Alice, a quantum computing researcher, wants to share an n-qubit density matrix ρ called the *message state* with her collaborator Bob. Eve, their unscrupulous competitor, can potentially intercept the state Alice sends, and will scoop them if ρ is unencrypted. Is it possible for Alice to conceal the state ρ in a *cipher state* $\tilde{\rho}$, so that even if Eve obtains $\tilde{\rho}$ she learns nothing? Of course, the encryption should also be reversible so that Bob can decode it. Such a scheme is called a *quantum one-time pad*.

The simplest way to implement reversible transformations in quantum mechanics is with unitary operators. So, we can imagine that Alice picks some unitary $U \in \mathrm{U}(2^n)$, and applies it according to the usual rule for updating density matrices:

$$\rho \mapsto \tilde{\rho} = U\rho U^\dagger.$$

If Bob knows U, he can easily decode. But here's the kicker: if $\tilde{\rho}$ contains *absolutely no information* about ρ, that simply means it cannot change with ρ, i.e. $\tilde{\rho}$ is constant. Sadly, this is impossible!

Exercise 4. Verify that no unitary U maps arbitrary ρ to fixed $\tilde{\rho}$.

While encoding with a single unitary does not work, a more general possibility is a *probability distribution* over unitaries, (p_k, U_k). This is a special case of a general formalism called a *quantum channel*, which we won't define here. The corresponding cipher state is

$$\rho \mapsto \tilde{\rho} = \sum_k p_k U_k \rho U_k^\dagger.$$

To implement this, Alice and Bob meet beforehand and pick a *key* k randomly according to the distribution p_k. Alice later sends $\tilde{\rho}_k = U_k\rho U_k^\dagger$, which Bob can easily undo if he knows k. But the initial randomization step means that the whole density (over many iterations of the scheme) is described by the statistical admixture $\tilde{\rho}$. The specific state $\tilde{\rho}_k$ cannot be constant, but the whole density $\tilde{\rho}$ *can* be constant. Let us first determine what constant it must be!

Exercise 5. Show that, if $\tilde{\rho}$ is constant, then $\tilde{\rho} = \mathbb{I}_{2^n}/2^n$, the maximally mixed density matrix. *Hint.* Consider the image of the mixed density.

Given Exercise 5, our job is now to find a distribution (p_k, U_k) such that

$$\sum_k p_k U_k \rho U_k^\dagger = \frac{\mathbb{I}_{2^n}}{2^n}$$

for any n-qubit density ρ. (A distribution with this property is called a *1-design*, coming under the broader umbrella of *combinatorial designs*.) Our intuition from the single unitary case suggests this is impossible, but thankfully, that intuition is wrong! It turns out that the uniform distribution over the Pauli group works. Before we prove this, we need a technical result.

Exercise 6. The *Hilbert-Schmidt inner product* of $2^n \times 2^n$ matrices M_1, M_2 is

$$\langle M_1, M_2\rangle = \mathrm{tr}[M_1 M_2^\dagger].$$

(a) Argue that elements of the set $\hat{\mathcal{P}}_n$ are orthogonal with respect to this inner product and hence span the set of $2^n \times 2^n$ matrices.

(b) Explain how to expand an arbitrary n-qubit density ρ in terms of $\hat{\mathcal{P}}_n$, i.e.

$$\rho = \sum_{\alpha\beta} c_{\alpha,\beta} X^{(\alpha)} Z^{(\beta)}.$$

We now have assembled all the ingredients needed to bake a delicious quantum one-time pad. Choose a *uniform* probability distribution $p_k = (4 \cdot 2^{2n})^{-1}$ over the elements of $\mathcal{P}_n$. Surprisingly, the simplest probability distribution over the simplest nontrivial subgroup of $\mathrm{U}(2^n)$ works!

Exercise 7. You may wish to revisit Exercise 3, 6, and 7 before embarking.

(a) The average over the full Pauli group is slightly redundant. Demonstrate that

$$\frac{1}{4 \cdot 2^{2n}} \sum_{U \in \mathcal{P}_n} U \rho U^\dagger = \frac{1}{2^{2n}} \sum_{\alpha,\beta \in \{0,1\}^n} X^{(\alpha)} Z^{(\beta)} \rho Z^{(\beta)} X^{(\alpha)}.$$

(b) Show that, for fixed $\gamma \in \{0,1\}^n$,

$$\sum_{\alpha \in \{0,1\}^n} (-1)^{\alpha \cdot \gamma} = \prod_{i=1}^{n} \left[1 + (-1)^{\gamma(i)}\right] = \delta_{\gamma,0}.$$

(c) Using Exercises 3, 6 and 7(b), conclude that for an arbitrary ρ,

$$\frac{1}{4 \cdot 2^{2n}} \sum_{U \in \mathcal{P}_n} U \rho U^\dagger = \frac{\mathbb{I}_{2^n}}{2^n}.$$

Thus, averaging over the Pauli group provides unbreakable quantum security for our researchers Alice and Bob. As discussed in Exercise 7(a), the full Pauli group $\mathcal{P}_n$ is slight overkill. Instead, we can consider a uniform distribution over the elements $X^{(\alpha)} Z^{(\beta)}$ of $\hat{\mathcal{P}}_n$. There are 2^{2n} elements, with two classical bits $\alpha(i), \beta(i)$ for every qubit Alice wants to transmit to Bob. For encryption and decryption, up to n Paulis are required (if we allow Y), so the whole procedure is efficient.

We finish with a cute extension. Suppose that Bob is Alice's graduate student. Alice wants Bob to perform some Pauli operations on ρ, say $X^{(\gamma)} Z^{(\delta)}$, but without giving Bob access to the state (i.e. the key). At the same time, she wants to protect her state from Eve. Is it possible to farm out work to her graduate student but keep the state secret from *both* Bob and Eve?

Exercise 8. Alice sends $\tilde{\rho}_k$ to Bob without telling him the key. From the viewpoint of both Bob and Eve, the full admixture is $\tilde{\rho} = \mathbb{I}_{2^n}/2^n$. Show that if Bob applies a set of Paulis $X^{(\gamma)} Z^{(\delta)}$ to the density he receives, then returns it to Alice, he completes the required computation.

This property is called *homomorphic encyption.* Like a group homomorphism, the encryption is structure-preserving, allowing Bob to safely operate on Alice's data without the key. Everything ties back to group theory!

III. Parallel Parking

The Hadamard transform

Morally, quantum computing is powerful because of the parallelism afforded by superposition. Thus, a common first step in most quantum algorithms is to prepare a uniform superposition of states. Let's see how this is done.

Exercise 1. In a sentence, apply Hadamards everywhere. But we should check this works!

(a) Suppose we have n qubits in the $|1\rangle$ state, i.e. the state $|1\rangle^{\otimes n}$. Show that applying a Hadamard to each produces

$$H^{\otimes n}|0\rangle^{\otimes n} = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle.$$

(b) Explain why this is precisely an even superposition,

$$\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle.$$

Hint. Use induction, or take the inner product with $\langle x'|$ for $x' \in \{0,1\}^n$.

We can modify this easily to prepare more interesting superpositions.

Exercise 2. The *Hadamard transform* applies Hadamards everywhere, but to arbitrary computational basis states. Let's see what happens.

(a) Show that for a single qubit, with $x \in \{0,1\}$, we can write the action of a Hadamard as

$$H|x\rangle = \frac{|0\rangle + (-1)^x|1\rangle}{\sqrt{2}}.$$

(b) For two qubits in the computational basis, $|x_1\rangle$ and $|x_2\rangle$, $x_1, x_2 \in \{0,1\}$, verify that

$$\begin{aligned} H^{\otimes 2}|x_1\rangle|x_2\rangle &= \frac{|00\rangle + (-1)^{x_2}|01\rangle + (-1)^{x_1}|10\rangle + (-1)^{x_1+x_2}|11\rangle}{\sqrt{2^2}} \\ &= \frac{1}{\sqrt{2^2}} \sum_{y \in \{0,1\}^2} (-1)^{x_1y_1 + x_2y_2}|y\rangle. \end{aligned}$$

(c) Let $x \cdot y = x_1y_2 + \cdots + x_ny_n$ denote the bitwise inner product for $x, y \in \{0,1\}^n$.

Extend your result from (b) to give

$$H^{\otimes 2}|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y\in\{0,1\}^n} (-1)^{x\cdot y}|y\rangle.$$

Hint. Again, you can either use induction (with base case (a) or (b)) or directly compute the inner product with $\langle y'|$.

(d) Applying $H^{\otimes n}$ is called the *Hadamard transform*. Consider the superposition

$$|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{y\in\{0,1\}^n} (-1)^{f(y)}|y\rangle$$

where f is any binary function $f : \{0,1\}^n \to \{0,1\}$. Show that the Hadamard transform of this state is

$$H^{\otimes n}|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{y,z\in\{0,1\}^n} (-1)^{f(y)+y\cdot z}|z\rangle,$$

and deduce from the special case $f(y) = y \cdot x$ that the Hadamard transform is self-inverse.

Since $H^2 = I$, it follows immediately that $H^{\otimes n}$ is self-inverse, so part (d) is really just a sanity check for our superposition formula. The Hadamard transform is closely related to the *Quantum Fourier Transform*, one of the central tools in quantum algorithms, to be discussed later.

Oracles

In a sense, it is "obvious" that quantum parallelism makes quantum computers more powerful than classical computers. Using a Hadamard transform, I can prepare a superposition of 2^n states, and by applying a single n-bit unitary U, perform exponentially many classical computations in one fell swoop! But linearity giveth and linearity taketh away. When we measure at the end of the day, we cannot learn $U|x\rangle$ for each $x \in \{0,1\}^n$. Instead, we just get a single number, and it is random. The subtlety and art of quantum computing lies in learning how to use the exponential power of quantum parallelism without throwing everything away when we make our final measurement.

To begin with, let's see how to use parallelism in the "obvious" way mentioned above. Suppose we have a binary function on n-bit strings, $f : \{0,1\}^n \to \{0,1\}$. We can define a gate U_f which acts on $n+1$ bits. The first n bits store the argument, and the last bit stores the answer:

$$U_f|x,y\rangle = |x, y \oplus f(x)\rangle.$$

This is unitary, and in fact $U_f^2 = \mathbb{I}$. Of course, U_f itself might be very hard to construct, but for our purposes, we will simply regard it as a black box or *oracle*. Using U_f is called an *oracle query*. Since the input qubits stores the query, we call it the *query register*.

Exercise 3. Here is our first little mini-algorithm: apply the Hadamard transform, then make an oracle query. As usual, we translate these words into math.

(a) Show that

$$U_f|x\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)=(-1)^{f(x)}|x\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right).$$

(b) Prepare the state $|0\rangle^{\otimes n}|1\rangle$. The state $|\psi_f\rangle$ is the result of applying the Hadamard transform and then making an oracle query U_f. Confirm that $|\psi_f\rangle$ takes the form

$$|\psi_f\rangle=U_fH^{\otimes(n+1)}(|0\rangle^{\otimes n}|1\rangle)=\frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}|x\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right).$$

So, we can perform an exponential number of classical calculations $f(x)$ in the "obvious" way. The problem, now, is how to extract anything useful from this, since the information is stored in phases.

If we measure the input n qubit, we will learn only a single phase $f(x)$. This is not particularly impressive. However, if we place some *global* constraints on f, involving *all* the values $f(x)$, it is plausible that by some clever manipulations we can extract global information about $|\psi_f\rangle$ from a single measurement. We will give three algorithms for this below. These constitute strong theoretical evidence that quantum computers can outperform classical computers.

Deutsch-Jozsa algorithm

You have already seen Deutsch's algorithm in lectures. The *Deutch-Jozsa algorithm* is a simple extension to the n-bit case we are considering. First, we will make the global constraint (technically called a *promise*) that the function f is either *constant* or *balanced*: the function is always the same (constant), or 1 half the time and 0 the rest.

Exercise 4. The Deutsch-Josza algorithm is simply to Hadamard transform the parallelized state, query the oracle, then measure the query register.

(a) Apply the earlier result 2(d) to conclude that

$$H^{\otimes n}|\psi_f\rangle=\frac{1}{2^n}\sum_{x,z\in\{0,1\}^n}(-1)^{f(x)+x\cdot z}|z\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right).$$

(b) Show that if we measure the query register, the state $|0\rangle^{\otimes n}$ is returned with probability $|\mathcal{A}_0|^2$, where

$$\mathcal{A}_0=\langle 0|^{\otimes n}\frac{1}{2^n}\sum_{x,z\in\{0,1\}^n}(-1)^{f(x)+x\cdot z}|z\rangle=\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}.$$

(c) Argue that if f is constant, then $\mathcal{A}_0=1$ and you are guaranteed to return $|0\rangle^{\otimes n}$. If it is balanced, show $\mathcal{A}_0=0$ and hence we cannot return $|0\rangle^{\otimes n}$.

In other words, when we look at the query register, we can tell whether f is constant or balanced depending on whether we observe $|0\rangle^{\otimes n}$ or not. This algorithm is *exact*, and uses only a single evaluation of f. In contrast, the best deterministic classical algorithm requires us to check at least half of the arguments, or $O(2^{n/2})$ function queries! It looks like we have an exponential speedup. Unfortunately, this is a bit misleading. A slightly fairer comparison is to the *randomized* classical algorithm. Like Deutsch-Jozsa, it takes a constant number k of function queries, with probability of success increasing exponentially with k. Deutsch-Jozsa is still faster, but it is really only a "constant" speedup.

Bernstein–Vazirani algorithm

We can keep playing this game, coming up with a global promise about f, and making a clever measurement to determine some associated global information. The *Bernstein–Vazirani algorithm*, for instance, promises that f is a "secret dot product", $f(x) = x \cdot s$ for some secret $s \in \{0,1\}^n$. As with Deutsch-Jozsa, we parallelize, Hadamard transform, query, then measure.

Exercise 5. Use Exercise 2(d) to show s is returned with probability $|\mathcal{A}_s|^2 = 1$.

In other words, the query register contains our answer exactly after a single query. The best classical algorithm requires n queries, on the "basis" strings $b_i = 0^i 1 0^{n-i-2}$, with

$$f(b_i) = s \cdot b_i = s_i.$$

So we reconstruct the "secret" s from the n function calls $f(b_i)$. This is a linear improvement on the best classical algorithm, which is beginning to look like progress. But it's still not exponential.

Simon's algorithm

Based on the preceding examples, it's easy to generalise to a function $f : \{0,1\}^n \to \{0,1\}^n$, and an oracle acting on $x, y \in \{0,1\}^n$ as

$$U_f|x, y\rangle = |x, y \oplus f(x)\rangle,$$

where $y \oplus f(x)$ is bitwise as usual. We will prepare a state $|\Psi_f\rangle$ slightly different from the above.

Exercise 6. Prepare the all-zero state $|0\rangle^{\otimes n}|0\rangle^{\otimes n}$. The parallelized state $|\Psi_f\rangle$ is, as before, prepared by Hadamard transforming the query register and making an oracle query. Show that

$$|\Psi_f\rangle = U_f(H^{\otimes n} \otimes I_n)|0\rangle^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle|f(x)\rangle.$$

Once again, to do something useful, we need to make a global promise about f so a single measurement gives nontrivial global information. *Simon's algorithm* starts with the promise that there is a secret string $s \in \{0,1\}^n$ such that, for $x \in \{0,1\}^n$,

$$f(x) = f(y) \iff y = x \oplus s.$$

The algorithm proceeds as before, but with a subtle difference.

Exercise 7. Like the previous algorithms, we Hadamard transform the query register. The subtlety is that before, and afterwards, we measure the *output* register, rather than the query register. Let's see what that buys us.

(a) First, measure the output register in $|\Psi_f\rangle$, and suppose the outcome is $|w\rangle$. Explain why the query register is in the state

$$\frac{|x\rangle + |x \oplus s\rangle}{\sqrt{2}}.$$

for some $x \in \{0,1\}^n$.

(b) Now Hadamard transform the query register. Show that it is in the state

$$\frac{1}{\sqrt{2^n}} \sum_{z \in \{0,1\}^n} \left(\frac{(-1)^{x \cdot z} + (-1)^{(x+s) \cdot z}}{\sqrt{2}} \right) |z\rangle.$$

(c) Finally, measure the output register. Argue that the only terms $|z\rangle$ with nonzero amplitude $\mathcal{A}_z$ satisfy $z \cdot s = 0$.

We have learnt something nontrivial about s, namely that it is orthogonal to z. If we repeat the circuit multiple times, we will gradually winnow down which subspace it is in, until we can uniquely determine s. The number of queries needed is, on average, $O(n)$, since each independent z cuts the space in half, so we go from the whole space of 2^n points to 2^1 after acquiring $n-1$ independent vectors z. What is the best classical algorithm?

Exercise 8. Consider $f : \{0,1\}^n \to \{0,1\}^n$ with the Simon promise, $f(x) = f(y)$ iff $y = x \oplus s$ for a "secret" s. Since there is no other structure at hand, the best approach is to randomly guess x and y until we find a "collision", $f(x) = f(y)$. This is called the "birthday problem". Explain why this algorithm takes $O(\sqrt{2^n})$ random guesses.

The best randomized classical algorithm is exponential, while Simon's algorithm is $O(n)$. Thus, we have our first example of an exponential quantum speedup! Unfortunately, the problem is rather contrived and useless, which led to reviewers initially rejecting Simon's paper. Scott Aaronson explains what happened next:

> *So the story goes that Simon wrote a paper about this theoretical black-box problem with an exponential quantum speedup, and the paper got rejected. But there was one guy who was like, "Hey, this is interesting." He figured that if you changed a few aspects of what Simon was doing, you could get a quantum algorithm to find the periods of periodic functions, which would in turn let you do all sorts of fun stuff. That guy was Peter Shor.*

Aaronson is alluding to *Shor's algorithm*, a famous quantum algorithm which factorizes numbers as a special case. It is most definitely useful! Together, these examples strongly suggest that quantum computers provide complexity-theoretic leverage over their classical cousins, not only in Simon's contrived black box problem, but the real world.

IV. Fourier and Fourierer

This assignment introduces the *Quantum Fourier transform (QFT)* for qudits, and describes how to implement it quickly on a quantum computer. This is our first (and perhaps only) practically useful exponential speedup over classical computers!

Position and momentum

Consider a d-dimensional Hilbert space $\mathcal{H}_d$. We will label the computational basis $|n\rangle$, $n = 0, \ldots, d-1$. To connect to our intuitions from ordinary quantum mechanics, we can view this standard basis as eigenvectors of a position operator, $X|n\rangle = x_n|n\rangle$. We could array these on a line, but a more natural choice is to array positions at equal intervals on the *unit circle*. In other words, we will choose $x_n = \omega^n$, where $\omega = e^{2\pi i/d}$ is a primitive d-th root of unity. Thus, we define a position operator

$$X = \begin{bmatrix} 1 & 0 & & \cdots \\ 0 & \omega & & \cdots \\ & & \ddots & \\ 0 & \cdots & & \omega^{n-1} \end{bmatrix}.$$

This is also called the *clock* matrix, since it tells the time on the unit circle.

Given this choice of representation, it's natural to seek eigenvectors of a momentum operator P. Put differently, we would like to look for objects which are eigenvectors of translation, or rather, rotation around the circle. Let P implement ccw rotations by $2\pi/d$, taking $P|n\rangle = |n+1\rangle$. Then

$$P = \begin{bmatrix} 0 & 1 & 0 & \cdots \\ 0 & 0 & 1 & \cdots \\ & & \ddots & \\ 1 & \cdots & & 0 \end{bmatrix}.$$

This is sometimes called the *shift* matrix.

Exercise 1. Although canonical commutation relations cannot hold for any choice of X and P in a finite-dimensional Hilbert space, we can nevertheless show these operators do not commute.

(a) In ordinary quantum mechanics, we have the canonical commutation relations $[X, P] = i\hbar\mathbb{I}$. Prove that there are no finite-dimensional operators which satisfy this. This is the wrong thing to look for with qudits!

(b) Show that $XP = \omega PX$, so these measurements do not commute.

(c) Check that, for $d = 2$, X and P *anticommute*, and relate them to Pauli matrices. (Our choice of label X is unfortunate in this context.) In a sense, (b) simply generalizes this familiar anticommutation relation.

For our next trick, we determine the eigenvectors of P.

Exercise 2. Write an arbitrary vector $|\chi\rangle = \chi_n|n\rangle$, using Einstein summation notation to sum over n.

(a) Suppose $P|\chi\rangle = \lambda|\chi\rangle$ for eigenvalue λ. Argue that $\chi_{n-1} = \lambda\chi_n$, where n is taken modulo d.

(b) Set $n = d$ in (a) to conclude that $\lambda^d = 1$. This means λ is a d-th root of unity.

The previous exercise gives a set of d linearly independent, normalized eigenvectors of P, one for each choice of $\lambda = \omega^{-s}$:

$$|\chi^s\rangle = \frac{1}{\sqrt{d}}\omega^{sn}|n\rangle. \tag{1}$$

The state $|\chi^s\rangle$ has "momentum" ω_d^{-s}. Although it is guaranteed by various linear algebra theorems, let's check this gives an orthonormal basis.

Exercise 3. Check that

$$\langle\chi^t|\chi^s\rangle = \delta_{st}.$$

Hint. Recall the geometric sum

$$1 + r + \cdots + r^p = \frac{1 - r^{p+1}}{1 - r}.$$

We can plonk these momentum eigenstates into the columns in a large matrix,

$$W_d = \left[\left|\chi^0\right\rangle, \left|\chi^1\right\rangle, \ldots, \left|\chi^{d-1}\right\rangle\right].$$

This is called the *Walsh-Hadamard matrix*, and indeed, it is a generalization of the Hadamard matrix, as you can easily check.

Exercise 4. In this exercise, we'll consider a qubit, i.e. $d = 2$.

(a) Define the clock and shift matrix in the computational basis.

(b) Check that the states $|\pm\rangle = (|0\rangle \mp |1\rangle)/\sqrt{2}$ are indeed eigenvectors of the shift matrix, with square roots of unity as eigenvalues.

(c) Verify that $W_2 = H$ is the usual Hadamard matrix.

Since the basis is orthonormal, the matrix is unitary, with $W_d W_d^\dagger = W_d^\dagger W_d = \mathbb{I}$, or

$$\langle\chi^t|\chi^s\rangle = \delta_{ts}, \quad |\chi^s\rangle\langle\chi^s| = \mathbb{I}.$$

We can use these to perform a change of eigenbasis from X to P. This change of eigenbasis is called the *discrete Fourier transform (DFT)*.

Exercise 5. Consider an arbitrary $|\psi\rangle = a_n|n\rangle = A_s|\chi^s\rangle$. Using the unitarity of the Walsh-Hadamard matrix, confirm that

$$a_n = \frac{1}{\sqrt{d}}\omega^{ns}A_s, \quad A_s = \frac{1}{\sqrt{d}}\omega^{-ns}a_n.$$

The DFT is a *passive* transformation, in the sense that it leaves the vector alone, but relates the coefficients in two different bases. In quantum computing, we only have access to a single basis: the computational basis. So, instead of looking at the same vector differently, we need to actively *change* to achieve anything useful. This active transformation is called the *quantum Fourier transform (QFT)*, defined on the computational basis by

$$\text{QFT} : |n\rangle \mapsto |\chi^n\rangle = W_d|n\rangle,$$

and extended to all states by linearity.

Exercise 6. Show that the QFT replaces the coefficients a_n with their Fourier duals, i.e.

$$\text{QFT}(a_n|n\rangle) = A_n|n\rangle.$$

Awesome powers

The DFT and QFT in a sense ask different questions. Since the DFT doesn't change the state, it really means *finding the coefficients* A_s. The QFT, on the other hand, changes the state, and in many cases (for instance, quantum computing), that's all we want. We don't need the individual coefficients! But in both cases, the computation can be sped up dramatically when the Hilbert space is a tensor product.

For simplicity, we will consider the case of λ qudits, i.e. $\mathcal{H} = \mathcal{H}_d^{\otimes\lambda}$, though the decomposition is more general. The Hilbert space $\mathcal{H}$ has dimension $d^\lambda = D$. Let $n_i = 0, \ldots, d-1$ label the X eigenbasis of the i-th tensor factor, and define the standard base-d expansion

$$N(n_0, n_1, \ldots, n_{\lambda-1}) := n_0 + n_1 d + \cdots + n_{\lambda-1}d^{\lambda-1} = \sum_{\ell=0}^{\lambda-1} n_\ell d^\ell. \tag{2}$$

In this section, we restore explicit summation for clarity. The number N runs from 0 to $D-1$, labelling a position eigenbasis of $\mathcal{H}$, with

$$|N\rangle = |n_0\rangle \otimes |n_1\rangle \otimes \cdots \otimes |n_{\lambda-1}\rangle.$$

Let ω_D, ω_d denote respective primitive roots, and $|\chi^S_{(D)}\rangle, |\chi^s_{(d)}\rangle$ position eigenvectors, in the tensor power and its factors. Let's see how they're related!

Exercise 7. As in (1), let

$$|\chi^S_{(D)}\rangle = \frac{1}{\sqrt{D}}\omega_D^{NS}|S\rangle.$$

(a) With N given by (2), show that

$$\langle N|\chi^S_{(D)}\rangle = \prod_{\ell=0}^{\lambda} \frac{1}{\sqrt{d}} \exp\left(2\pi i S n_\ell d^{\ell-\lambda}\right).$$

(b) Conclude that the momentum eigenvectors factorize, with

$$|\chi^S_{(D)}\rangle = \bigotimes_{\ell=0}^{\lambda-1} |\chi^{Sd^{\ell-\lambda+1}}_{(d)}\rangle.$$

(c) We can simplify further by expanding S itself in base d. Define

$$S := s_0 + s_1 d + \cdots + s_{\lambda-1}d^{\lambda-1} = \sum_{\ell=0}^{\lambda-1} s_\ell d^\ell \tag{1}$$

$$S^{(j)} := s_0 d^{\ell-\lambda} + s_1 d^{1+\ell-\lambda} + \cdots + s_{\lambda-\ell-1}d^{-1} = \sum_{\ell=0}^{\lambda-j-1} s_\ell d^{j+\ell-\lambda}. \tag{2}$$

Show that, in base d,

$$S^{(j)} = 0.s_{\lambda-j-1}s_{\lambda-j-2}\ldots s_0.$$

(d) Using the previous question, verify that

$$\exp\left(2\pi i S n_\ell d^{\ell-\lambda}\right) = \exp\left(2\pi i S^{(\ell)} n_\ell d^{\ell-\lambda}\right)$$

and conclude as a result that

$$|\chi^S_{(D)}\rangle = \bigotimes_{\ell=0}^{\lambda-1} |\chi^{S^{(\ell)}}_{(d)}\rangle = \frac{1}{\sqrt{D}} \bigotimes_{\ell=0}^{\lambda-1} \sum_{n_\ell=0}^{d-1} e^{2\pi i S^{(\ell)} n_\ell}|n_\ell\rangle. \tag{3}$$

It seems innocuous, but this little factorization leads to a dramatic quantum speedup. To calculate the QFT or the DFT using a classical computer, without exploiting the factorization, we must compute the D coefficients A_S. Each of these requires $O(D)$ arithmetical operations, since

$$A_S = \frac{1}{\sqrt{D}} \sum_{N=0}^{D-1} \omega^{-NS} a_N.$$

All told, this takes $O(D^2) = O(d^{2\lambda})$ operations. This is exponential! Let's how much better we can do using tensor factorization.

Exercise 8. The *Fast Fourier Transform (FFT)* exploits (3) to compute the DFT quickly for a tensor power.

(a) Argue that each component A_S takes only $O(d\lambda)$ basic arithmetic operations to evaluate.

(b) Conclude that the FFT takes $O(d^{\lambda+1}\lambda)$ steps. This is still exponential, but much less so!

There are other variants of the FFT which can speed things up further with additional assumptions about the numbers, but in general, this is the best we can do on a classical or quantum computer if we want to find the individual Fourier coefficients.

Short circuits

The QFT does not find individual coefficients. Instead, it rotates the computational basis elements into the momentum eigenbasis. On a classical computer, this is just as hard as a DFT, since we need to compute the individual coefficients A_S as before. But on a quantum computer, we can exploit the fact that superposition and tensor products are baked into the hardware to sidestep the DFT and directly compute the QFT. As we will show shortly, there is a quantum circuit which takes a mere $O(\lambda^2)$ steps! This is a genuine exponential speedup over the best known classical algorithm, and the most spectacular result in quantum computing, from a complexity-theoretic view. (Simon's Problem also has exponential speedup, but unlike the QFT, is not useful.)

Let's define the problem a little more carefully. First of all, we are going to *fix* d, the qudits that form the basis of our computer, and let the number λ vary, with $D = d^\lambda$. We will construct the Walsh-Hadamard matrix W_D that performs the QFT in $O(\lambda^2)$ steps using two-qudit gates we will define in a moment. A special case is the usual qubit circuit for performing the QFT, where $d = 2$. Let's introduce our gates. In addition to the qudit Walsh-Hadamard matrix, W_d, first define the *bunched* position operator

$$X_d^{(t)} := X_d^{d^{1-t}} = \mathrm{diag}\left(1, e^{2\pi i/d^t}, e^{2\pi i \cdot 2/d^t}, \ldots, e^{2\pi i(d-1)/d^t}\right).$$

This is "bunched" since it takes the d points which were originally equally spaced on the unit circle, and bunches them onto an arc of angular size $2\pi d^{1-t}$. These provide a natural set of operators for doing the base d expansions we need for (3).

Exercise 9. Let S and $S^{(\ell)}$ be as above. Show that

$$\left(X_d^{(\lambda-\ell)}\right)^{s_0}\left(X_d^{(\lambda-\ell-1)}\right)^{s_1}\cdots\left(X_d^{(2)}\right)^{s_{\lambda-\ell-2}} = X_d^{S'} \tag{4}$$

where $S' = S^{(\ell)} - s_{\lambda-\ell-1}d^{-1}$.

There is a controlled version of bunching, defined by

$$CX_d^{(t)}|n\rangle|m\rangle = \left(X_d^{(t)}\right)^m|n\rangle.$$

For qubits, this reduces to $m = 0$ (off) and $m = 1$ (on), but for qudits we get a power. As usual, in a circuit we place a black dot on the control qudit. With our gate set in hand, we can finally define the circuit. The basic idea of the algorithm for the QFT is simple. We will define a circuit which acts on the basis elements as follows:

$$|s_\ell\rangle \mapsto \frac{1}{\sqrt{d}}\sum_n e^{2\pi S^{(\ell)}n}|n\rangle. \tag{5}$$

Exercise 10. Using (3) and linearity, confirm that a circuit performing (5) computes the QFT of $|\psi\rangle \in \mathcal{H}_D$.

Our circuit is simple and elegant, but works slightly better if we first reverse the order of qudits, so $|s_\ell\rangle \mapsto |s_{\lambda-\ell-1}\rangle$. We will omit this step in our circuit. As usually happens when we want to exploit quantum parallelism, we then apply the Walsh-Hadamard matrix to each qudit, taking a position eigenvector to a momentum eigenvector. So

$$|s_\ell\rangle \mapsto |s_{\lambda-\ell-1}\rangle \mapsto W_d|s_{\lambda-\ell-1}\rangle = |\chi_{(d)}^{\lambda-\ell-1}\rangle.$$

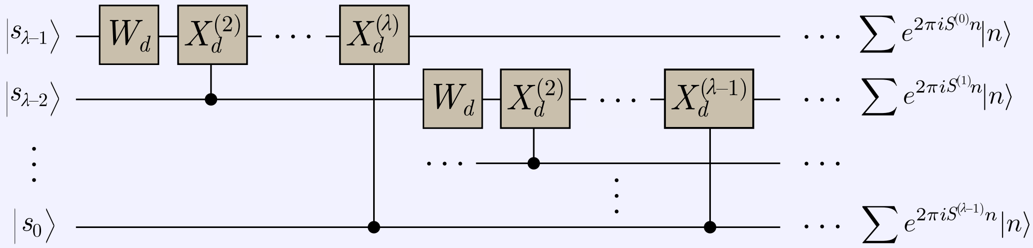
We now just apply the bunching relation (4) to obtain the correct factor for (3).

Exercise 11. Let's put everything together and blow classical computers out of the water!

(a) Using Exercise 9, prove that

$$X_d^{S'}|\chi_{(d)}^{\lambda-\ell-1}\rangle = \frac{1}{\sqrt{d}}\sum_n e^{2\pi S^{(\ell)}n}|n\rangle.$$

(b) Conclude from Exercises 9 and 10 that the following circuit does the QFT:



(c) Show that the number of gates in this circuit is $\lambda^2/2$. One subtlety we have ignored is the number of swaps required. Assuming that swapping adjacent qudits is an elementary operation, show that the total number of gates is λ^2.

Most textbooks compress all this into a couple of pages of dense algebra. Hopefully, we've unzipped the inner workings of the QFT a little.

V. Pete Breaks the Internet

The goal of this (long and optional) assignment is to understand how to make and break the cryptographic systems underlying internet security. In particular, we will see what real-world implications a fully-fledged quantum computer would have, and use this as a launch point to discuss the ethics of quantum computing.

1. Making public key cryptosystems

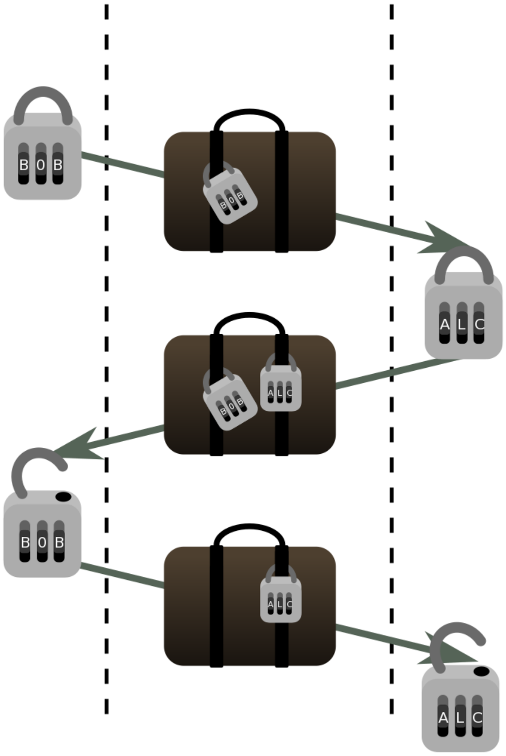
Suppose Alice and Bob want to exchange sensitive information over an insecure channel, e.g. the internet. The final member of the cryptographic trinity is *Eve*, a malicious third party who wants to compromise Alice and Bob's communications. If Alice and Bob can meet in person beforehand, they can agree on a shared scheme for *encrypting* their information. For instance, if they know in advance the length of messages they want to exchange, e.g. a credit card number, they can share a *one-time pad* consisting of random bits. This is provably unbreakable.

Of course, this scheme is hugely constraining. In the world of internet commerce, Alice and Bob will often be on other sides of the world, so meeting in person is impractical. If Bob from Boston wants to buy a hand-knit yak wool sweater from Alice in Amdo, and wants to pay her electronically, how can he ensure his details aren't compromised by Eve? The ingenious solution is *public key cryptography.* Public key cryptography was invented by Diffie and Hellman [1] in the late 70s, and the first practical scheme developed by Diffie and Hellman and independently by Merkle [2]. Many of the ideas were actually discovered earlier by Ellis, Cocks and Williamson [3], but because they were working for Britain's codebreaking agency GHCQ, their work was classified.

1.1. The suitcase analogy

Before launching into the mathematical details, there is a simple analogy which conveys the main idea. Instead of sending credit card data to Alice, suppose Bob wants to send cash in a suitcase. He first attaches a combination lock, then sends the suitcase to Alice. Remember that Alice and Bob have not met, and do not have a more secure communication channel over which Bob can safely tell her the combination. Instead of opening the suitcase, Alice attaches *her own combination lock*, and sends it back to Bob. Bob now removes his combination lock, and sends the suitcase back to Alice. By removing her own lock, Alice can finally retrieve Bob's money, thereby concluding the transaction! We draw the exchange below.

Alice and Bob never needed to meet or exchange information on an insecure channel; they just needed their own (hopefully secure) locks. A *public key cryptosystem (PKC)* works in much the same way. Metaphorically, a PKC provides suitcases and combination locks that Alice and Bob can set themselves. But it makes the suitcase and lock out of math, which is cheaper, quicker to send, and harder to break! In the next section, we will introduce the first PKC, developed by Diffie, Hellman and (independently) Merkle.



1.2. Diffie-Hellman-Merkle (DHM) and discrete logarithms

The original PKC is called *Diffie-Hellman-Merkle (DHM) key exchange*, since it only allows Alice and Bob to share a *key* rather a full message. The key is a shared, secret number which can be used for a different encryption method, e.g. the combination of a combination lock. In mathematical terms, Alice and Bob publicly announce a large prime p and some integer g modulo p. These are known to everyone, including Eve. This is like announcing the suitcase and brand of combination lock.

To put a "lock" on the suitcase, Bob picks a large power g^b, modulo p. (This can be done efficiently using modular exponentiation, covered elsewhere.) The number b is his "combination", and the result is a singly-locked "suitcase":

$$B \equiv g^b \bmod p.$$

This number B is called Bob's *public key.* Bob sends this to Alice, who raises the result to her own power, a:

$$K \equiv B^a \equiv g^{ab} \bmod p.$$

Alice has added her on lock onto Bob's lock, which is like an empty suitcase with both locks, mathematically represented by K. They now repeat the protocol, but in reverse order. Alice raises g to the power a, yielding her public key

$$A \equiv g^a \bmod p.$$

Alice sends A to Bob, who raises it to his secret power b:

$$K \equiv A^b \equiv g^{ab} \bmod p.$$

In the suitcase analogy, both Alice and Bob have empty, doubly-locked suitcases. The analogy is a little strained, since we need to assume that the order of locks in the "double-locking" procedure is irrelevant, and that the locks can then be used as an input to another locking scheme! But mathematically, it's much simpler: Alice and Bob just share a number K.

Let's think about the security of this procedure. Eve knows g and p, since they were publicly announced, and can potentially intercept the "singly-locked suitcases" $A \equiv g^a$ and $B \equiv g^b$. If she can quickly take logarithms modulo p, also called *discrete logarithms*, she can compute a and b from these. Using modular exponentiation, she then immediately calculates the key $K \equiv g^{ab}$. So the security of key exchange boils down to the following question: how hard are discrete logarithms? We start exploring this question in the first exercise.

Exercise 1. Let's start with the most terrible algorithm.

(a) Let $\mathbb{F}_p^*$ denote the multiplicative group modulo p. Show that $g^p \equiv 1 \bmod p$ for any element g.

(b) Argue that $\mathbb{F}_p^*$ is cyclic, i.e. there is a multiplicative generator g such that $\mathbb{F}_p^* = \{1, g, g^2, \ldots, g^{p-1}\}$. Such a g is called a *primitive element. Hint.* Use the fact that every finite Abelian group is a product of cyclic groups. If you like elementary proofs, check out this cute collection.

(c) Show that, for primitive g, the "discrete logarithm" $\log_g h$ of any element $h \in \mathbb{F}_p^*$ is well-defined, provided it takes values in $\mathbb{Z}_{p-1}$.

(d) From part (c), deduce that the complexity of taking discrete logarithms via brute force is $\mathcal{O}(p)$.

Taking discrete logarithms in the group $\mathbb{F}_p^*$ can be done in $\mathcal{O}(p)$ time. So, if Alice and Bob want to protect themselves from Eve, they should choose an exponentially large prime p, say $p \sim 2^k$, so the prime takes k bits to specify. Brute force enumeration will then be exponential in k, $\mathcal{O}(2^k)$. We will discuss better algorithms for taking discrete logs below. You might wonder if we can jerry rig the DHM scheme to send messages. *ElGamal encryption*, a full-fledged PKC invented by Tahar ElGamal [5] in 1985, does just this.

Exercise 2. Bob wants to send Alice his credit card details so she can charge him for the yak wool sweater. As before, the numbers g and p are made public, and now Bob chooses a one-off key or *nonce* k, for this message only, while Alice uses her private key a to create a public key $A \equiv g^a$ as before. Alice sends her public key to Bob, while Bob encodes his plaintext as a number $m \in \mathbb{F}_p^*$. He then computes two encrypted *ciphertexts*, which he sends to Alice:

$$c_1 \equiv g^k \bmod p, \quad c_2 \equiv mA^k \bmod p.$$

Show that $m \equiv c_1^{-a} c_2$ mod p, so Alice can recover Bob's message.

Although we have discussed protocols for finite fields modulo p, we can generalize key exchange and ElGamal to an arbitrary group G. Many exotic cryptographic methods, e.g. elliptic curve cryptography, just replace $\mathbb{F}_p^*$ with a fancy group, in the hopes the discrete logarithm problem will be harder to solve. While encrypting is more or less the same, *breaking* these cryptosystems does involve new and fancy math!

1.3. One-way functions and trapdoors

Let's consider DHM more abstractly. We can view powers in a group G as a function

$$\mathrm{pow} : G \times \mathbb{Z} \to G$$

which can be efficiently computed using modular exponentiation. If we define $f(x) = \mathrm{pow}(g, x) = g^x$ as a function of the power alone, the discrete logarithm problem is simply to invert f. (By "invert", we mean find a single element of $f^{-1}(h)$.) DHM works because the discrete logarithm problem is hard for appropriately chosen G, i.e. f is difficult to invert. The best known algorithm for $G = \mathbb{F}_p^*$ is subexponential, but still much faster than any polynomial, as we discuss in Section 2.1.

Functions that are easy to compute but hard to invert are called *one-way functions*. The discrete exponential is one example, but *all* modern cryptography is based on (apparently) one-way functions of one form or another. I say "apparently" since no one is sure one-way functions actually exist! It's easy prove that you can compute something efficiently, but very hard to prove you cannot compute it efficiently, since you need to somehow consider every possible algorithm. This is the same problem underlying P vs. NP. In fact, proving the existence of a one-way function *implies* that $\mathsf{P} \neq \mathsf{NP}$, so if you find one, you get a million bucks! Internet commerce is one giant bet that these complexity classes are distinct.

Although DHM is ingenious, it involves two rounds of communication and only computes a key. Diffie and Hellman [1] observed that constructing a PKC is simple if you have a *trap door function*. A trap door function is a special one-way function which can be easily inverted if you have "trap-door" information. This is like a combination lock, where the combination is the trap-door information. Suppose Alice publicly advertises her trap door function f but not the trap-door information. Bob forms his plaintext message m, computes the ciphertext $c = f(m)$, and sends it to Alice. Alice uses her trap-door information to easily recover m. Eve, on the other hand, is stuck with the computationally infeasible task of inverting $f(k)$ without the trap-door information.

This sounds great, but trap door functions are much harder than one-way functions! Despite their efforts, Diffie and Hellman were unable to find a single plausible candidate. Before moving on, let's see why discrete logarithms do *not* provide a trap door.

> ***Exercise 3.*** Let $A \equiv g^a \bmod p$ be Alice's public key, as above. Explain why $f(k) = A^k$ is not a trap door function. Even though she knows a, Alice is no better off than Eve!

1.4. The Rivest-Shamir-Adleman (RSA) cryptosystem

Enter Ron Rivest, Adi Shamir and Leonard Adleman (RSA), who two years after Diffie and Hellman's famous paper, found the first plausible trap-door function [7]. RSA has a colourful history. River and Shamir were computer scientists, whose role in the collaboration was to generate wacky trap-door candidates. Adleman, the mathematician of the trio, would shoot them down. In April 1977, the trio spent Passover evening together, drinking liberal quantities of Manischewitz wine and parting ways after midnight. After getting home, Rivest couldn't sleep, and lay awake mulling the problem. Suddenly, he was struck by inspiration, and wrote the bulk of the paper that very night! (Once again, poor old Clifford Cocks at GHCQ was "post-scooped", devising a similar system in 1973. His scheme was only declassified in 1997, long after the RSA trio had become rich and famous.)

To understand Rivest's revelation, let's reverse the roles of x and g in the discrete logarithm problem and consider *roots* of h:

$$x^g \equiv h \bmod p.$$

Now the power is known, but not the g-th root x. For a prime p, finding roots is easy.

Exercise 4. Let's find discrete roots $x^e \equiv h$ modulo p, assuming that e and $p-1$ are relatively prime.

(a) Show that e is coprime with $p-1$ just in case it has a multiplicative inverse modulo $p-1$, $ed \equiv 1 \bmod (p-1)$.

(b) Using part (a), conclude that x has the unique solution

$$x \equiv h^d \bmod p.$$

Hint. Recall *Fermat's Little Theorem*, $a^{p-1} \equiv 1 \bmod p$ for any integer a.

This does not sound like a promising approach to a trap-door function. However, now consider replacing p with a *product* of distinct primes or *semiprime* $N = pq$. To find roots, we need a generalization of Fermat's little theorem called *Euler's little theorem*.

Exercise 5. Let $\ell = \operatorname{lcm}(p-1, q-1)$ and suppose a is relatively prime to $N = pq$. Prove that

$$a^\ell \equiv 1 \bmod N.$$

Hint. The integer $a^\ell - 1$ is divisible by both p and q if and only if it is divisible by pq.

We can now generalize Exercise 4 to roots modulo $N = pq$.

Exercise 6 (roots modulo a semiprime). Suppose that e is relatively prime with $N' = (p-1)(q-1)$.

(a) Show that $de \equiv 1 \bmod N'$.

(b) Suppose h is relatively prime to N. Conclude using Euler's little theorem that $x^e \equiv h$ modulo N has unique solution

$$x \equiv h^d \bmod N.$$

This will form the basis of the RSA trap door function. Suppose Alice chooses large primes p and q, and computes their product $N = pq$. She makes N publicly available, as well as the index e, so the public keys are (N, e). The trap door function is

$$f(x) = x^e \bmod N,$$

with trap-door information (private key) either the factors p and q, N', or the inverse d. Bob can choose a plaintext m, compute $c = f(m)$, and send it to Alice. Alice can easily determine m using Exercise 6. And voila! That's RSA. We will refer to such a system by the triple (N, e, d).

What about security? Eve's task, if she wishes to learn m, is to solve

$$m^e \equiv h \bmod N$$

without knowing the trap-door information. This is called the *RSA problem*, and it is believed to be hard. The simplest way to solve it is to directly obtain the trap-door information, i.e. factorize N. It's possible there is some sneaky way to get m without factorizing N, but no one knows!

First, we'll discuss brute-force factorization. Since N is a semiprime, all we need to do is find a single prime factor. If both factors are larger than $\sqrt{N}$, then the product is larger than N, which is impossible, so the simplest way to proceed is to test *primes below* $\sqrt{N}$. How many are there? To estimate, we need a Big Theorem, but like discrete logarithms, our algorithm is still exponential when $N \sim 2^k$ is a product of two $k/2$-bit primes.

Exercise 7. The *Prime Number Theorem (PNT)* is one of the crowning achievements of 19th century mathematics. Let $\pi(n)$ be the number of primes less than or equal to n. The PNT states that, for large n,

$$\pi(n) \sim \frac{n}{\log n}.$$

It was proved independently by Jacques Hadamard and Charles Jean de la Vallée Poussin in 1896.

(a) Show that, if we select a number randomly between 1 and N, it is prime with (approximate) probability $1/\log n$.

(b) Assume that checking for divisibility costs nothing. Using the PNT, show that the brute-force approach to factorizing $N \sim 2^k$ is exponential in k, with time complexity $O\left(k^{-2}2^{k/2}\right)$.

Before finishing the section, we provide a second method called *order finding* for breaking RSA which does not yield trap-door information.

Exercise 8. There is another way to break RSA called *order finding*. Eve finds m^e mod N, and knows both e and N since they were publicly announced. Instead of factoring N, she can determine the *order* of the element m^e, i.e. find an index r such that $(m^e)^r = m^{er} \equiv 1 \bmod N$. Let's see how she can find m.

(a) We can find r such that $(m^e)^r = 1$ just in case m^e and N are relatively prime. If they are not, explain how finding the common factor (which can be done efficiently using Euclid's algorithm) breaks RSA.

(b) Using Exercise 5, show that if m^e and N are relatively prime, r divides N'.

(c) For the RSA protocol, we assume that e is relatively prime with N'. Combining this with the previous exercise, conclude that e is relatively prime to r, and hence there is a multiplicative inverse $ed' \equiv 1 \bmod r$.

(d) Finally, show that $(m^e)^{d'} \equiv m \bmod N$.

Eve learns the inverse d' modulo r, rather than the trap-door inverse d modulo N', but that is enough! The bottleneck here is the problem of order-finding. On a classical computer, the naive algorithm is simple to go through the possible orders and see if they give 1. This is linear in N, and hence exponential in the number of bits k for $p, q \sim 2^{k/2}$. This is the best classical algorithm we know!

2. Breaking the internet

We have learned how to make a PKC. We will now discuss how to *break* a PKC with a classical computer. Rather than discussing the mathematical details, we will simply present the resource scaling and consider the real-world implications. If we compromise DHM or RSA, what else goes wrong?

2.1. Transport Layer Security

PKCs play an important role in the cryptographic architecture of the internet. A central example is *digital signatures*. Suppose Bob is an internet user wanting to communicate securely with a server Alice, e.g. to send credit card details. These transmissions should of course be encrypted to prevent Eve from stealing the data. But Alice and Bob should check that the messages are coming from the right people! A scheme for authenticating the identity of communicating parties is called a *digital signature*.

RSA provides a simple example. Suppose that Bob and Alice use an RSA system with public parameters (N, e) and private key d ($de \equiv 1$ modulo N') in Alice's possession. Bob wishes to check that Alice is who she says she is. Alice sends a plaintext message m (which in itself contains no sensitive information) along with the signature $\sigma \equiv m^d$ modulo N. Bob knows e, and can simply check that $\sigma^e \equiv m$. Problem solved? Not exactly. If Eve has enough control over the channel Alice and Bob are using, she can *pose as Alice* and set up an (N, e, d) of her choice. She can, of course, easily authenticate that she was the one who chose N! This is called the *man-in-the-middle* attack.

To guard against these attacks, Alice and Bob must have recourse to a trusted third party: *Vera*. Vera is a publicly known authority who can somehow verify the identity of Alice and sign things for her. This is implemented using a protocol called *Transport Layer Security (TLS)*. Without entering into the details, it loosely works as follows. Rather than appending her own signature to a nonce message, Alice sends a *certificate signing request* to Vera. Vera magically verifies Alice is not Eve, and signs the *TLS certificate*. Bob checks the certificate comes from Vera (who has her own digital signature), and inside, finds Alice's public key. Of course, Eve can try posing as Vera, but it will typically be much harder to fake being a trusted public authority than a private Alice or Bob in the all-to-all chaos of internet traffic! (Note that, when browsing, you can wrap everything in a TLS layer by appending "s" to "http".)

Exercise 9 (fake IDs). Consider ways that Eve could trick Bob into communicating with her, using legitimate TLS certificates.

2.2. Breaking public key cryptosystems classically

RSA gives a simple trapdoor function and hence an elegant digital signature scheme. For this reason, it remains the most popular method for providing TLS certificates. As a result, the security of RSA does indeed have major implications for internet security in general. We gave some naive algorithms for breaking RSA above, but the best classical algorithms make clever use of *number sieves*. We won't go into the details, but the *general number sieve* algorithm scales as

$$O\left(e^{1.9(\log N)^{1/3}(\log\log N)^{2/3}}\right).$$

Currently, RSA Security recommends a key size of $k = 2048$ bits, i.e. $N \sim 2^{2048}$. Let's see why!

Exercise 10. The Folding@home distributed computing network commands a flabbergasting 2.3 exaFLOPS, meaning it can perform 2.3×10^{18} floating-point operations per second. Let's suppose the network devotes itself to using number seive methods to factorize large numbers.

(a) Estimate how long it would take Folding@home to factorize: (i) a 512-bit key, $N \sim 2^{512}$; (ii) a 1024-bit key; (ii) a 2048-bit key.

(b) Moore's law predicts that computing power doubles every 1.5 years. Roughly how long do you expect $k = 2048$ to remain secure?

While ubiquitous in digital certificates, RSA is almost never used for the subsequent secure communication. Creating a new RSA cryptosystem for each communication is too expensive, but if Alice and Bob reuse their system, and the private keys are subsequently compromised, then *all prior communications are compromised*. Clearly, for each secure exchange, we should generate unique keys once identities have been authenticated. Such a system will then possess *forward secrecy*.

This is where DHM key exchange (and its elliptic curve cousin) gets used in practice. After Vera certifies Alice and Bob, they generate a shared key, and use that as input for one of the many symmetric encryption methods (usually some flavour of AES). Since a new key is generated for each exchange, the system, called *Diffie-Hellman ephemeral* or DHE, is in principle forward secret provided the keys used are large and chosen independently for each exchange. But in practice, keys are often reused, and solving the discrete logarithm problem for specific prime moduli p can compromise large parts of the internet [6]. Many SSH, VPN and mail servers, as well as the majority of websites, use fixed-group DHE. Among other recommendations, [6] suggests using larger primes, e.g. 2048 bits, and moving from standard DHM to the elliptic curve variant, which is not susceptible to number sieve methods.

The state-of-the-art classical algorithm for solving the arbitrary discrete logarithms in $\mathbb{F}_p^*$ also uses number sieves, with a scaling similar to factorization:

$$O\left(e^{1.9(\log p)^{1/3}(\log\log p)^{2/3}}\right)$$

Since the discrete logarithm case is simpler, we will sketch the related (not quite state-of-the-art) *index calculus* algorithm. For more details, see [4].

Exercise 11. Let p be prime and g a primitive root of $\mathbb{F}_p^*$. We will pick a value L and for all primes $\ell \le L$, compute $\log_g \ell \bmod p$. This will let us solve the discrete logarithm problem for arbitrary group elements!

(a) Suppose we have computed these values, and want to find $\log_g h$ for some $h \in \mathbb{F}_p^*$. Consider the quantities

$$h \cdot g^{-n} \bmod \mathrm{p},$$

where $n = 1, 2, \ldots$. Increment n until $h \cdot g^{-n}$ has no prime factors greater than L. Show that

$$h \cdot g^{-n} = \prod_{\ell \leq L} \ell^{j_\ell} \quad \Longrightarrow \quad \log_g h = n + \sum_{\ell \leq L} j_\ell \log_g \ell.$$

This step of the process is much faster than computing $\log_g \ell$, so we turn to this bottleneck next!

(b) To compute the small prime logarithms, $\log_g \ell$, randomly generate a set of indices i, compute g^i, and factorize. If g^i contains prime factors larger than L, discard it, but otherwise it can be expressed

$$g^i = \prod_{\ell \leq L} \ell^{r_\ell(i)} \quad \Longrightarrow \quad i = \sum_{\ell \leq L} r_\ell(i) \log_g \ell.$$

Sketch how you can obtain the small prime logarithms $\log_g \ell$ once we have $\pi(L)$ such numbers, where $\pi(x)$ is the number of primes less than or equal to x.

(c) Numbers with no prime factor above L are called *L-smooth.* To determine the running time of the index calculus method, we need to determine how long it takes to find $\pi(L)$ L-smooth numbers using the randomly chosen g^i. Explain qualitatively how it might be possible that the number of draws required is approximately independent of L.

(d) As suggested in the previous question, for an arbitrary N, the number of draws required to produce $\pi(L)$ L-smooth powers g^i mod p is

$$\mathcal{O}\left(e^{\sqrt{2 \ln p \ln \ln p}}\right),$$

independent of L. Deduce that this algorithm for the discrete logarithm problem has complexity $\mathcal{O}\left(e^{\sqrt{2 \ln p \ln \ln p}}\right)$, and show that this is slower than the state-of-the-art number sieve method.

2.3. Quantum quandaries

We have seen that, in order to safeguard PKCs and hence the internet from powerful classical computers, we simply need to increase key size. For the time being, $k = 2048$ bits for both RSA and DHM seems reasonable, though key size needs to keep pace with Moore's law and should increase in future. But quantum computers change things rather dramatically. In comparison to the near-exponential number sieve methods, Shor's algorithm can factor $N = pq$ in

$$O\left(\log^3 N\right)$$

steps. This is shockingly fast. In fact, this scales the same way as operations with the private key! Similarly, we can solve the discrete logarithm problem for $\mathbb{F}_p^*$ in $O(\log^2 p)$ steps.

But there are some subtleties lurking behind this simple scaling. First, the constants sitting out the front are large. Second, in order to successfully run the algorithm, the quantum computer needs to be noise-tolerant. Finally, size matters: it takes $\sim 2 \log N$ qubits to store the number N and run the order-finding part of the algorithm. Thus, we need to build a fault-tolerant quantum computer with around 4000 logical qubits before we can break RSA or DHM with 2048-bit keys. See [11] for more on these practical considerations. But whenever it happens, the current TLS architecture will need to change when large-scale quantum computers arrive.

Exercise 12. In [11], Gidney and Ekerå propose that 20 million noisy qubits should be sufficient to solve RSA-2048 in 8 hours. Moore's law appears to apply to qubits as well, with processor size doubling every 1.5 years. The next few years should see the advent of noisy intermediate-scale quantum computing (NISQ), with 50–100 qubits.

(a) Assuming Moore's law holds, how long until we reach Gidney and Ekerå's threshold?

(b) Explain why scaling up the RSA key size will be impractical once large-scale quantum computers arrive. For instance, if private key operations take a millisecond for RSA-2048, and it takes 8 hours to break, what happens when we choose a key that takes 100 years to solve? Assume the asymptotic constants don't change.

Not only are RSA and standard DHE efficiently solvable by a quantum computer, but the elliptic curve variant as well. (Recall that Shor's algorithm generalizes to solving the hidden subgroup problem for abelian groups.) An immediate question, then, is how to protect internet security once large-scale quantum computing becomes a reality.

Exercise 13. Here is a little post-quantum tapas.

(a) Look up the *multi-prime* variant of RSA, where $N = pq$ is replaced by a product of three or more primes. How does it work? Is it secure against quantum computers?

(b) Research the *post-quantum TLS ciphersuite*. How does it achieve digital authentication and forward-secrecy? Is anyone using it? What are the performance tradeoffs, and are these likely to have any side effects?

(c) *Quantum key distribution (QKD)* is often touted as the quantum solution to the quantum problems posed by Shor's algorithm. Is it likely to be useful for internet security? If not, where is it likely to be useful?

The march of technology is exciting to watch from afar, and even more exciting to be part of. But it is worth pausing to consider the broader implications of the fun we are are having.

Exercise 14. In a blog post from 2019, Scott Aaronson dismisses concerns that quantum computing is an arms race. But is this too quick? Aaronson comments that "the morality. . . could strongly depend on whether the codebreakers are working for the good guys or the bad guys". Let's expand on this!

(a) With specific reference to quantum computing and internet security, explain how who has what technology when, and how they use it, can have major geopolitical implications. You may want to consider historical examples (e.g.

poor old Clifford Cocks, Bletchley Park, the Manhattan Project, etc).

(b) If you were a quantum computing researcher on the verge of a scalability breakthrough, what would be the most reponsible course of action? Should you talk to Verisign (real-world Vera) before publishing? The government? Post it on Reddit? What would it depend on? Think carefully about the likely impact.

(c) Imagine you are a policy advisor for the Canadian government. How would you advise them to proceed?

(d) Aaronson points out that large-scale quantum computing is likely to have many benefits, to materials science, medicine, and perhaps even climate change. But who benefits? Comment on social and economic factors which could determine this. It may be helpful to first answer this question for classical computers or the internet.

References

1. New directions in cryptography (1976), Whitfield Diffie and Martin Hellman.
2. Secure communications over insecure channels (1978), Ralph Merkle.
3. The possibility of secure non-secret digital encryption (1970), J. H. Ellis.
4. An Introduction to mathematical cryptography (2008), Jeffrey Hoffstein, Jill Pipher, Joseph H. Silverman.
5. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms (1985), Tahar ElGamal.
6. Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice (2015), David Adrian et al.
7. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems (1978), R.L. Rivest, A. Shamir, and L. Adleman.
8. Quantum algorithms for computing general discrete logarithms and orders with tradeoffs (2020), Martin Ekerå.
9. Understanding SSL/TLS (2008), J. K. Harris.
10. The state of factoring algorithms and other cryptanalytic threats to RSA (2013), Daniel J. Bernstein, Nadia Heninger, and Tanja Lange.
11. How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits (2019), Craig Gidney and Martin Ekerå.
12. The Impact of Quantum Computing on Present Cryptography (2018), Vasileios Mavroeidis, Kamer Vishi, Mateusz D. Zych, and Audun Jøsang.