Quantum Computing, Part 3

Andrew Glassner

Every now and then surprising new theories appear on the scientific stage that hold the promise of dramatic new technologies. Quantum computing is one of these. The ideas in this field radically change the way we think about computers and computing.

In my last two columns, I introduced the theory of quantum computing and presented its basic terminology and notation. In this installment, I'll wrap up my discussion of the subject by presenting some interesting quantum algorithms and then showing how quantum computing can change the world of cryptography, or the sharing of secrets.

Before we get going, I'll briefly recap some of the most important points from the last two columns. If you haven't read those columns, you might want to take a look at them because this summary will be more of a reminder than a tutorial. Toward the bottom of the review section, I'll introduce a few conventions that will be useful in this issue.

Our story so far

The quantum version of a bit is the *qubit*. We write a qubit's state as a *ket*, which is a two-element column vector of complex numbers. The traditional states are $|0\rangle$ and $|1\rangle$. A qubit can be in a *superposition* of states, when it's literally in two or more states at once. We write this superposition as a linear sum of the states: $\psi=a|0\rangle+b|1\rangle$, where a and b are complex numbers. When we observe a qubit, it's projected into one of the states allowed by its own structure and the measuring apparatus. Once projected, the qubit stays in that state until otherwise manipulated.

The probability of finding a superimposed particle in any given state is the square of the weight on that state. For ψ in the last equation, the probability of finding it in state $|0\rangle$ is $|a|^2$. By convention, we normalize the weights so that the sum of their squares is 1.

When we mix two states equally, we can write $\psi=a|0\rangle+a|1\rangle$. Normalization requires that $|a|^2+|a|^2=1$, which for a real value of a means $a=1/\sqrt{2}$. This normalization is so common that I will use the symbol $\sigma=1/\sqrt{2}$ in this column. I'll assume that all of our coefficients are real from here on out.

We can realize qubits in practice with a variety of quantum particles. For example, we can represent the two states $|0\rangle$ and $|1\rangle$ with two perpendicular polarizations of a photon.

We can create systems of multiple qubits, which are the quantum equivalents of classical registers. We assemble qubits into a quantum register with the *tensor product*, written $\otimes$. For example, given qubits α_0, α_1, and α_2, we can build a 3-qubit register $\alpha = \alpha_0 \otimes \alpha_1 \otimes \alpha_2$. We write the state of such a system as a ket with multiple elements, such as $|010\rangle$.

We can create *entangled pairs*, also called EPR pairs. These pairs are only permitted to exist in particular configurations, so if we measure one particle, the other instantly snaps into its required state no matter where it is. Therefore, if one particle of an entangled pair is on Earth and the other is on Saturn, at the precise moment when one of the particles is observed (and thus takes on one and only one state), the other particle is also instantly projected into its corresponding state. For example, if the pair is $\psi_0=|00\rangle+|11\rangle$ and we measure the first particle and find it to be $|0\rangle$, the second particle also instantly is projected into state $|0\rangle$. Another EPR pair is $\psi_1=|01\rangle+|10\rangle$. If we find the first particle in state $|0\rangle$, then the second particle instantly snaps into state $|1\rangle$.

Because column vectors represent quantum states, we can transform them with matrices. As we saw last issue, matrices can represent quantum gates that are the counterparts of all the traditional classical gates (such as AND and OR).

In addition to those gates, there are five common operations that we'll use this issue. The first four are

Errata

In Part 2 of this series (published in the September/October 2001 issue), a couple of errors slipped into Figure 7 (page 95).

In Figure 7a, the wire marked C_{in} has a dot where the left-most vertical line crosses it. That dot should be moved two wires to the right. Thus, the first Toffoli gate is controlled by inputs *a* and *b*, the second by *a* and C_{in}, and the third by *b* and C_{in}.

In Figure 7b, the vertical lines that form the Toffoli gates on the wire marked C_{in} shouldn't extend below the bottom of the open circles.

$$I: \begin{matrix} |0\rangle & \rightarrow & |0\rangle \\ |0\rangle & \rightarrow & |1\rangle \end{matrix} \quad = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$X: \begin{matrix} |0\rangle & \rightarrow & |1\rangle \\ |1\rangle & \rightarrow & |0\rangle \end{matrix} \quad = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

$$Y: \begin{matrix} |0\rangle & \rightarrow & -|1\rangle \\ |1\rangle & \rightarrow & |0\rangle \end{matrix} \quad = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

$$Z: \begin{matrix} |0\rangle & \rightarrow & |0\rangle \\ |1\rangle & \rightarrow & -|1\rangle \end{matrix} \quad = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

I is the *identity* transformation, which does nothing to its inputs. The other three transformations have names given them by convention and have nothing to do with the *X*, *Y*, and *Z* coordinate systems we know and love from 3D space.

Another common gate is the *Hadamard* gate, written *H*:

$$H: \begin{matrix} |0\rangle \rightarrow \sigma(|0\rangle + |1\rangle) \\ |1\rangle \rightarrow \sigma(|0\rangle - |1\rangle) \end{matrix} = \sigma\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

All five transforms are *unitary*. The definition of unitary for any matrix *M* involves *M* and its conjugate transpose M^*. It states that if *M* is unitary, then $MM^* = \pm I$. If *M* is made up of only real numbers, then M^* is just M^T, or the transpose of *M*. We represent these gates schematically with a simple box containing the gate's letter.

The Hadamard gate's utility is that it can take a qubit in a pure state such as $|0\rangle$ and turn it into a superposition of states. If we apply it to each qubit of an *n*-bit register, the result is a register that simultaneously represents 2^n different values. When we apply *H* to *n* qubits of a register simultaneously, we call it the *Walsh* (or *Walsh–Hadamard*) transformation on that register and write it as *W*.

Cloning

Cloning is a hot topic. To the news media, it's all about bringing dinosaurs back from the dead and creating designer babies. But in the quantum-computer world, the term cloning has a less controversial meaning.

Suppose that you have a qubit ϕ, and you'd like to make a few identical copies of it. That's no problem: measure ϕ to determine its state, and then make a bunch of new qubits in that state. But suppose instead that you want to make copies of ϕ without measuring it first. That is, you want to make more qubits in the same state as ϕ, but you don't want to determine the value of ϕ in the process. This would be helpful because you could make a few copies of ϕ and work with them while they were still in their superimposed state. If it took a long time to build up the state of ϕ, and you wanted to run several experiments, it would be handy to make some copies of this input so you wouldn't have to build it up from scratch each time.

Unfortunately, cloning isn't possible. You simply can't make a perfect copy of a quantum particle without first determining its state. The proof is short and sweet. We'll suppose that there is some unitary transformation *U* that clones, and then we'll see that this leads to a contradiction. I chose the letter *U* to remind us that the transform is unitary (recall that means $UU^* = \pm I$).

U works by taking in a 2-qubit register $|a0\rangle$ made up of an unknown qubit in state $|a\rangle$ and another qubit in state $|0\rangle$. *U* turns that second state into a copy of $|a\rangle$ without measuring $|a\rangle$. In symbols, we want *U* to produce the following:

$$U(|a0\rangle) = |aa\rangle$$

Let's imagine a second state $|b\rangle$, which is orthogonal to $|a\rangle$. We don't care about the particular choice of $|b\rangle$ as long as its inner product with $|a\rangle$ is 0. Symbolically, we want $\langle a|b\rangle = 0$. This is easily arranged for any $|a\rangle$. Because $U(|a0\rangle) = |aa\rangle$, we also have $U(|b0\rangle) = |bb\rangle$. Let's make a third state that is an equal combination of $|a\rangle$ and $|b\rangle$: $|c\rangle = \sigma(|a\rangle + |b\rangle)$. Because *U* clones any input, it will also clone this state: $U(|c0\rangle) = |cc\rangle$. As you're probably expecting, these conclusions aren't consistent, and will give us a contradiction. Let's see how.

First, let's write out the cloning of $|c\rangle$ by expanding out the value of $|c\rangle$ from its definition. We start by observing that

$$\begin{aligned} |c0\rangle &= |c\rangle \otimes |0\rangle \\ &= \sigma(|a\rangle + |b\rangle) \otimes |0\rangle \\ &= \sigma(|a0\rangle + |b0\rangle) \end{aligned}$$

Since *U* is linear, we can write

$$\begin{aligned} U(|c0\rangle) &= U(\sigma(|a0\rangle) + |b0\rangle) \\ &= \sigma((U|a0\rangle) + (U|b0\rangle)) \\ &= \sigma(|aa\rangle + |bb\rangle) \end{aligned}$$

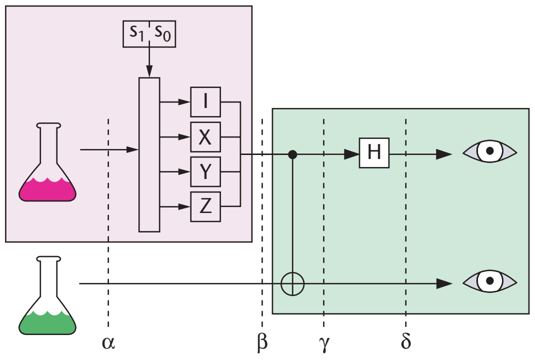
So far, so good. We have one way to express $U(|c0\rangle)$. We said earlier that $U(|c0\rangle) = |cc\rangle$, so let's expand that out and see what it looks like:

$$\begin{aligned} U(|c0\rangle) &= |cc\rangle \\ &= |c\rangle \otimes |c\rangle \\ &= \sigma(|a\rangle + |b\rangle) \otimes \sigma(|a\rangle + |b\rangle) \\ &= \sigma^2|aa\rangle + \sigma^2|ab\rangle + \sigma^2|ba\rangle + \sigma^2|bb\rangle \\ &= \sigma^2(|aa\rangle + |ab\rangle + |ba\rangle + |bb\rangle) \end{aligned}$$

Uh-oh. This last expression, which is a derivation of $U(|c0\rangle)$, is obviously not equal to $\sigma(|aa\rangle + |bb\rangle)$, which we just derived as another version of $U(|c0\rangle)$. That's our contradiction.

So this shows that at least one of our starting premises must be false. But our only starting premise was that there exists a unitary transformation *U* that clones. Therefore, there can't be any such transformation, and therefore we can't clone states.

You might be wondering if there's a nonunitary trans-



1 **Dense coding. Alice and Bob share an EPR pair, labeled α. The qubits are labeled from the top, so α_0 is in Alice's control, and α_1 is in Bob's. Alice's part of the process is in the red box. Alice takes the first 2 bits of her message and uses it to direct qubit α_0 into one of four transformations. She sends the resulting qubit to Bob. Bob's steps are in the green box. Bob applies a C_{not} gate to his 2-qubit system β creating γ. He measures γ_1 immediately and runs γ_0 through an *H* gate to create δ_0, which he measures.**

Table 1. Alice's encoding step for dense coding (see Figure 1).

s	Operation	β
00	$(I \otimes I)\alpha$	$\sigma(\lvert 00\rangle+\lvert 11\rangle)$
01	$(X \otimes I)\alpha$	$\sigma(\lvert 10\rangle+\lvert 01\rangle)$
10	$(Y \otimes I)\alpha$	$\sigma(-\lvert 10\rangle+\lvert 01\rangle)$
11	$(Z \otimes I)\alpha$	$\sigma(\lvert 00\rangle-\lvert 11\rangle)$

formation that clones. Although I haven't proven it, all the transforms that we use in quantum computing are unitary transformations, so there's no escape hatch—we simply can't create perfect clones.

Although we can't clone an unmeasured particle, don't forget that we have no trouble making copies of a known state; just measure it and manufacture copies. We also might be able to make an imperfect cloning device. It's possible we could find an operation *U′* that creates clones almost, but not quite, all the time. If we were willing to tolerate that ambiguity or could devise algorithms that were statistically insensitive to such errors, we might have a practical tool for cloning states even though we're denied perfection.

Dense coding

Let's look at a neat use of entanglement to speed up communication. Suppose that anytime you wanted to send a piece of information to a friend, you could arrange to use some other information shared long beforehand, so that when it's time to send your message you only have to send half of it. That would be pretty cool, and it's what we get from the quantum technique called *dense coding*.

The basic idea is this: Alice wants to send 400 classical bits worth of information to Bob—by classical bits I mean our old friends 0 and 1 from today's standard computing. Of course, Alice might be able to compress her message and save a few bits, so for the sake of the discussion let's assume that Alice has applied every compression trick in the book and really needs to send all 400 bits to Bob. In classical communication, there's no alternative but to send all 400 bits.

Let's assume that Alice and Bob knew several years ago that someday Alice would want to send Bob a message. So they went to their local quantum-particle store together and bought 200 EPR pairs. Each particle of each pair comes in its own little bottle, so Alice and Bob can easily identify particle 0 of pair 0, particle 1 of pair 0, particle 0 of pair 1, and so on. Alice takes home one particle from each pair and Bob takes home the other. Each locks up their particles in a safe and forgets about them.

Years pass, and now it's time for Alice to send Bob her 400-bit message. Using dense coding, and because Bob holds the other particle of 200 entangled pairs, Alice can transmit her message by manipulating her 200 particles and then sending only those 200 qubits to Bob.

To see how this works, let's break Alice's original message up into sequential pairs of classical bits. Each pair gets handled the same way, so we'll just pretend that Alice only wants to send a 2-bit message *s*, composed of classical bits s_1 and s_0, which represent the decimal numbers 0 to 3. These are the first 2 bits in Alice's message.

Associated with this first pair of classical bits will be the first EPR pair. Because there are two particles in the EPR pair, they can represent any one of four states. What will happen is that Alice will encode her particle in such a way that when she sends it to Bob, he can combine it with the particle he already has to determine which of the four states Alice intended. Figure 1 illustrates the idea. I've labeled the particles at different points in the process so that we can identify them easily.

Alice starts with her classical bits s_0 and s_1 in hand and then unlocks the safe and takes out her bottle marked "EPR Pair 0, Qubit 0." This is her half of the EPR pair marked α in the figure. She knows that Bob will eventually take out his bottle marked "EPR Pair 0, Qubit 1" later on to decode the message. As in Figure 1, I'll write the original state of this 2-qubit entangled EPR pair as $\alpha = \sigma(\lvert 00\rangle+\lvert 11\rangle)$.

Alice starts out by using the 2-bit value of *s* to perform one of four transformations on her qubit of α.

Even though Alice controls only the first qubit of α, it's implicitly part of an entangled 2-qubit system. So I'll write her operation as a transformation to the entire state. In practice, that means that if Alice applies transformation *A* to qubit α_0, she's implicitly applying the identity transformation *I* to qubit α_1, which of course doesn't change it at all.

Alice uses the 2-bit value of *s* to choose a transform from the following list: *I*, *X*, *Y*, *Z*. It's important that Alice and Bob agree on the order of the operations in this list; the particular order I'm using here is the conventional one. So for example, if $s = 01$, Alice applies $X \otimes I$ to α. The result is β, which is a modification of the original entangled pair α where Alice has transformed the first qubit (β_0), but the second qubit (β_1) is unaffected.

Let's track the resulting system values for each of Alice's four choices. Starting with $\alpha=\sigma(\lvert 00\rangle+\lvert 11\rangle)$, Alice creates β by modifying the first qubit according to her choice of transformation (see Table 1). Notice that β is just α with an altered first qubit.

Having applied this operation to the pair, Alice sends her qubit to Bob via traditional channels—for example, she gives it to a courier who drives it over to Bob's house. The point is that she only sends one qubit to Bob.

Now that Bob has both qubits in his hand, he applies a C_{not} gate, transforming the system from state β to γ. To see what these four expressions look like, let's write out the application of C_{not} to each of the four possibilities for β (see Table 2). I've written out the 2 qubits of γ independently in the table's right-most two columns. The important thing to notice here is that the two qubits of γ are now unentangled: Bob can measure either one without affecting the other.

Bob starts by measuring the second qubit, γ_1. If this is $|0\rangle$, Bob knows the original 2-bit message that Alice wanted to send was either 0 or 3. Similarly, if γ_1 is $|1\rangle$, Bob knows the input was either 1 or 2. To disambiguate between these, Bob now applies H to the first qubit to create δ_0 (see Table 3). The table column labeled δ_0 is the result of simplifying the value of $H(\gamma_0)$. By measuring this qubit, Bob can now completely identify the 2-bit message that Alice wanted to send. If δ_0 is in state $|0\rangle$, then Alice's original 2-bit message was either 0 or 1. Similarly, if δ_0 is in state $|1\rangle$, then Alice's message was either 2 or 3.

Combining δ_0 with γ_1, Bob now knows exactly which of the four states Alice's original 2-bit signal represents. Alice sent Bob only a single qubit, but she managed to communicate her 2-bit message.

The conclusion is that dense coding lets us transmit a message of n classical bits by sending over only $n/2$ qubits. The technique depends on an preexisting reservoir of entangled particles that the correspondants share.

Dense coding has practical value when the costs of transmitting information is high or the communications link is slow.

One place where both of these criteria hold is in space. Electrical power is precious aboard a spacecraft, and every bit of information that's radioed back to a receiving station uses up some of that energy. To complicate things, sometimes a spacecraft only has limited windows of opportunity in which to transmit and receive data, so the information must flow as quickly as possible. Dense coding helps a spacecraft save time and energy by requiring only half the number of classical bits to be transmitted. This halves the transmission time and thus halves the power drain: a double win!

One drawback to dense coding as I've presented it here is that one needs to actually physically send and receive qubits. This might be particularly tricky for a spacecraft. Besides the inconvenience, the costs of a delivery mechanism can steal away the gains offered by dense coding. The next section describes a way to send the state of a quantum particle from one place to another without the need for bicycle messengers.

Teleportation

Closely related to dense coding is a technique called *teleportation*. This name was perhaps an optimistic description for this process, because it conjures up *Star Trek* images of people beaming up from far-away planets. In today's teleportation, we aren't really beaming dogs, people, or even cantaloupes from one place to another. Rather, we send a quantum particle's state from one place to another without looking at it.

Table 2. Bob's first encoding step for dense coding.

β	$\gamma = C_{not}(\beta)$	γ_0	γ_1
$\sigma(\|00\rangle+\|11\rangle)$	$\sigma(\|00\rangle+\|10\rangle)$	$\sigma(\|0\rangle+\|1\rangle)$	$\|0\rangle$
$\sigma(\|10\rangle+\|01\rangle)$	$\sigma(\|11\rangle+\|01\rangle)$	$\sigma(\|1\rangle+\|0\rangle)$	$\|1\rangle$
$\sigma(-\|10\rangle+\|01\rangle)$	$\sigma(-\|11\rangle+\|01\rangle)$	$\sigma(-\|1\rangle+\|0\rangle)$	$\|1\rangle$
$\sigma(\|00\rangle-\|11\rangle)$	$\sigma(\|00\rangle-\|10\rangle)$	$\sigma(\|0\rangle-\|1\rangle)$	$\|0\rangle$

Table 3. Bob's second decoding step for dense coding.

γ_0	$H(\gamma_0)$	δ_0
$\sigma(\|0\rangle+\|1\rangle)$	$\sigma(\sigma(\|0\rangle+\|1\rangle)+\sigma(\|0\rangle-\|1\rangle))$	$\|0\rangle$
$\sigma(\|1\rangle+\|0\rangle)$	$\sigma(\sigma(\|0\rangle-\|1\rangle)+\sigma(\|0\rangle+\|1\rangle))$	$\|0\rangle$
$\sigma(-\|1\rangle+\|0\rangle)$	$\sigma(\sigma(\|0\rangle-\|1\rangle)-\sigma(\|0\rangle+\|1\rangle))$	$\|1\rangle$
$\sigma(\|0\rangle-\|1\rangle)$	$\sigma(\sigma(\|0\rangle+\|1\rangle)+\sigma(\|0\rangle-\|1\rangle))$	$\|1\rangle$

That last statement might come as a surprise because I proved earlier that we can't clone a quantum particle. That is, we can't make a copy of it without looking at it. Teleportation doesn't violate that proof, but it provides a sneaky way around it. We can reproduce the particle at another location without measuring it first, but we must destroy the original in the process. That means we can't actually clone it, in the sense of making an identical one in addition to the first, but we can recreate it elsewhere. This is probably the sense in which the *Star Trek* metaphor actually could apply: If you step on the transporter pad, you disappear from the ship and reappear on the planet. To create the new you, the old you is necessarily taken apart; there can only be one of you at a time. In some *Star Trek* episodes, something goes wrong with the teleporter and they end up with two or more copies of the teleported person. In our terminology, that would be cloning, so that particular horror scenario isn't possible in quantum teleportation.

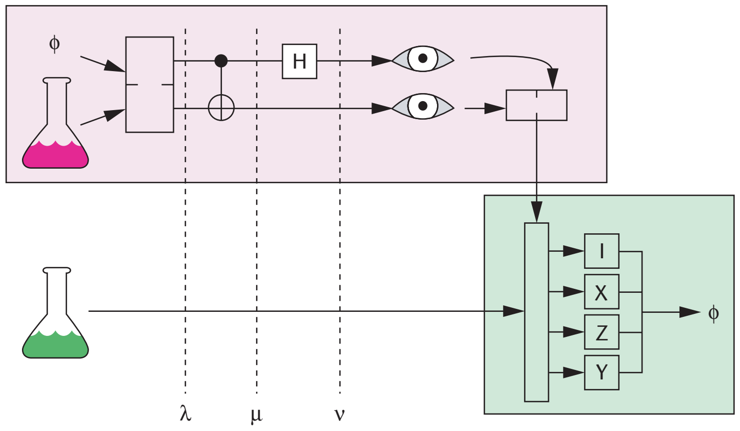
Like dense coding, an EPR pair α that Alice and Bob share makes teleportation possible. Suppose that Alice has a single qubit ϕ that she wants to send to Bob, but she doesn't know what state ϕ is in and she doesn't want to measure it. We can write the qubit ϕ as

$$\phi = a|0\rangle + b|1\rangle$$

and although Alice doesn't know a or b, she wants Bob to end up with a new qubit that is in the same state as ϕ. The process will follow roughly the same stages as dense coding in reverse, as Figure 2 (next page) shows.

To teleport ϕ to Bob, Alice creates the tensor product of the unknown qubit ϕ with an entangled pair $\alpha=\sigma(|00\rangle+|11\rangle)$:

$$\begin{aligned}\phi \otimes \alpha &= a|0\rangle \otimes \sigma\left(|00\rangle + |11\rangle\right) + b|1\rangle \otimes \sigma\left(|00\rangle + |11\rangle\right) \\ &= \sigma\left(a|000\rangle + a|011\rangle + b|100\rangle + b|111\rangle\right)\end{aligned}$$



2 Teleportation. Alice and Bob share an EPR pair. Qubits are again labeled downward, with 0 at the top. Alice's steps are in the red box. Alice combines α_0 with her input ϕ to create a system called λ. She executes a C_{not} to make μ and then applies an H gate to μ_0 to create ν. Alice measures ν_0 and ν_1 to get a value from 0 to 3, which she sends to Bob using two classical bits. Bob uses this 2-bit message to apply one of four transformations to his half of the EPR pair, creating a new particle in state ϕ.

Table 4. Teleportation (see Figure 2).

Bits from Alice	ν_2	Operation	Result
$\|00\rangle$	$a\|0\rangle+b\|1\rangle$	I	$a\|0\rangle+b\|1\rangle$
$\|01\rangle$	$a\|1\rangle+b\|0\rangle$	X	$a\|0\rangle+b\|1\rangle$
$\|10\rangle$	$a\|0\rangle-b\|1\rangle$	Z	$a\|0\rangle+b\|1\rangle$
$\|11\rangle$	$a\|1\rangle-b\|0\rangle$	Y	$a\|0\rangle+b\|1\rangle$

Now Alice has a 3-bit quantum register, created out of ϕ and α; let's call it λ.

Alice now applies $(C_{not} \otimes I)$ to λ to get μ, which will change the second qubit depending on the status of the first:

$$\begin{aligned}\mu &= \left(C_{\text{not}} \otimes I\right)\lambda \\ &= \sigma\left(a\left|000\right\rangle + a\left|011\right\rangle + b\left|110\right\rangle + b\left|101\right\rangle\right)\end{aligned}$$

Next, Alice applies H to the first qubit of μ to get ν:

$$\begin{aligned}\nu &= \left(H \otimes I \otimes I\right)\mu \\ &= \sigma^2\Big(a\left(\left|000\right\rangle + \left|011\right\rangle + \left|100\right\rangle + \left|111\right\rangle\right) \\ &\qquad + b\left(\left|010\right\rangle + \left|001\right\rangle - \left|110\right\rangle - \left|101\right\rangle\right)\Big) \\ &= \sigma^2\Big(\left|00\right\rangle\left(a\left|0\right\rangle + b\left|1\right\rangle\right) + \left|01\right\rangle\left(a\left|1\right\rangle + b\left|0\right\rangle\right) \\ &\qquad + \left|10\right\rangle\left(a\left|0\right\rangle - b\left|1\right\rangle\right) + \left|11\right\rangle\left(a\left|1\right\rangle - b\left|0\right\rangle\right)\Big)\end{aligned}$$

The result of all this work is that Alice now has a 3-qubit system in state ν, which is a superposition of four, equally probable states.

Alice's last step is to measure these first 2 qubits. Because ν_1 is entangled with qubit 1 of α (now named ν_2), when Alice measures ν_1, she's implicitly projecting Bob's particle as well. Alice has now caused Bob's qubit to be projected into one of four possible states.

Alice's measurement of the first 2 qubits identifies four possible states. Alice can encode this result using two classical bits to create a binary number from 0 to 3. Alice sends these two classical bits to Bob over classical channels, such as radio or the Internet.

An important point to notice is that when Alice measured the first 2 qubits of ν, she caused both qubits to be projected into some particular state. The first qubit of ν came from her original qubit ϕ. So when Alice measured the first 2 qubits, ν_0 got projected into some particular state, and lost its superimposed qualities. In other words, she lost ϕ forever. This is why teleportation isn't cloning; the input qubit ϕ gets destroyed in the necessary measurement step.

Now Bob is ready to recreate the original ϕ that Alice wanted to send him. He uses the two classical bits that Alice sends to apply one of four transforms to his qubit (ν_2), according to Table 4. In other words, by applying the appropriate transformation to the third qubit of ν, Bob has managed to reconstruct the original qubit ϕ.

Success!

The cost of teleportation is in the setup: creating and sharing an entangled pair, sending two classical bits, and destroying the quantum particle on Alice's end. The technique's value is that Bob can make a copy of ϕ without requiring Alice to measure it. Note that strictly speaking Alice and Bob don't have to share the EPR pair before the transmission because Alice can send the quantum particle ν_2 to Bob along with her two classical bits.

Teleportation gives us a way to communicate with spacecraft using dense coding after all. As long as we send it up with a big supply of entangled particles, we can get a lot of efficient communication.

Quantum-key distribution

Cryptography is an important subject for any new communication and computing technique, and quantum computing is no different. Effective cryptography is a huge field, which I won't even start to summarize here. Rather, I'll just focus on one of the exciting developments that are on the horizon as a result of quantum computing.

To set the stage, if Alice and Bob want to share a secret message, they need at least three things: the secret message in readable form (called the *plaintext*), a medium for communicating (the *channel*), and a way to make the subject unintelligible to anyone else but lets Bob recover the plaintext (the *cryptographic method*).

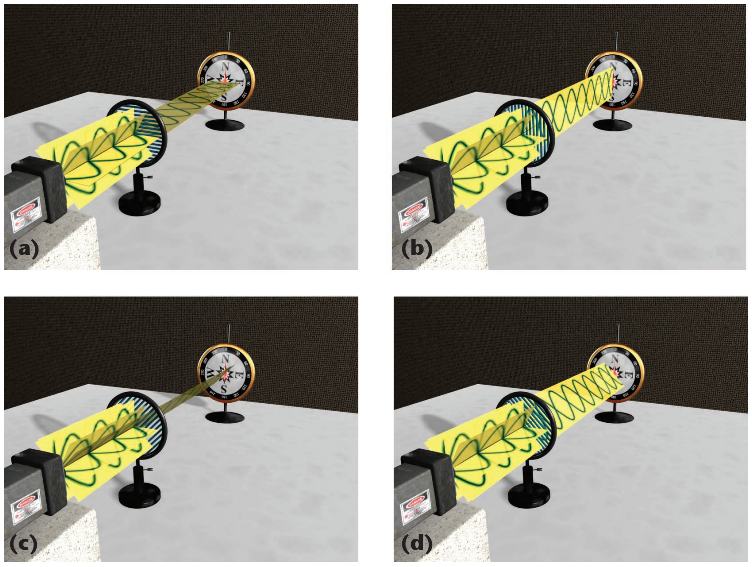
Perhaps the safest of all cryptographic methods is called the *one-time pad*. I'll describe it here briefly because it will give us a point of reference for the next discussion.

At some point, Alice and Bob get together and create

P	0	0	1	1	0	1	1	0	0	1
K	1	1	0	1	0	0	1	1	0	0
$P \oplus K$	1	1	1	0	0	1	0	1	0	1
$(P \oplus K) \oplus K$	0	0	1	1	0	1	1	0	0	1

3 ***P*** **is the plaintext message.** ***K*** **is the key. To create the ciphertext, compute** $P \oplus K$**. To recover** ***P*** **from** $P \oplus K$**, simply XOR it with** ***K*** **again.**

a pair of identical codebooks. In their simplest form, these codebooks are just big books with pages of random numbers. To send her secret information, Alice first writes out her message in plaintext. Then she looks at the first letter of her message and the first letter of the codebook. For a simple example, suppose the letter is A and the number in the codebook is 4. Alice adds four letters to A to get E, and that becomes the first letter of the secret message, called the *ciphertext*. In this way, Alice works her way through the plaintext letter by letter, using sequential entries from the codebook, creating the ciphertext. When she reaches the bottom of a page in the codebook she turns it over, and when a sheet has been used up on both sides she rips it out and burns it, moving on to the next page.



4 When Alice creates her photons, she controls their polarization. (a) $|\rightarrow\rangle$**: Setting** ***S*** **for a 0 bit. (b)** $|\uparrow\rangle$**: Setting** ***S*** **for a 1 bit. (c)** $|\nearrow\rangle$**: Setting** ***D*** **for a 0 bit. (d)** $|\nwarrow\rangle$**: Setting** ***D*** **for a 1 bit.**

When Bob receives the message, he simply follows the process in reverse, subtracting the codebook's numbers from each letter in the ciphertext to recover the plaintext. He too burns the codebook as he works through it. The technique is called the one-time pad because each entry in the codebook (or pad) is used only once and then destroyed.

This is obviously a simple example, but with well-designed codebooks, and algorithms for using them, this process is secure. Much of its power comes from the fact that the codebook is a one-time information source, so even if eavesdroppers somehow manage to crack a single message, they must start over again to crack the next one. The one-time pad lets the encrypted message be publicly communicated—it can be read over the radio or even printed in a newspaper. Without the codebook, the message is perfectly safe.

Despite its strength in protecting secrets, the logistics of the one-time pad make it difficult to use in practice. Perhaps the hardest part is getting together to share the codebooks and then keeping them secure until needed. Therefore, people have devised a wide range of alternative methods for creating, distributing, and recovering information in a codebook, also known as the *key* or the *one-time key*.

In digital computing, we often treat everything in sight as a simple string of bits. Think of the plaintext and the key as just long lists of bits, as in Figure 3. To encrypt the message, Alice just computes the XOR (written $\oplus$) of each plaintext bit with the corresponding bit from the key and sends the result; this is the ciphertext. Bob then does another XOR of the same key with the ciphertext to recover the plaintext.

The famous Enigma machine from World War II contained a complex mechanical device for creating an encryption key. The widely used commercial RSA system uses products of prime numbers to produce a key.

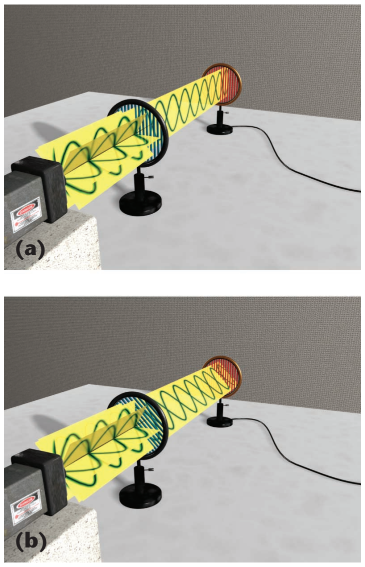
One of the toughest practical aspects of the one-time key is that both parties must meet ahead of time to exchange identical copies of the key and then keep them safe and secure. It would be great if there was a convenient way to create and exchange a one-time key when it was needed.

That's exactly what quantum computing gives us. Quantum-key distribution, or QKD, offers a way for two people to create a one-time key on demand in a safe and convenient way. They can even check to make sure that nobody has intercepted the key while it was being sent. Let's see how it works.

In this description, I'll use polarized photons as an example of a quantum particle. Recall from part 1 of this series that we can polarize a photon in either of two perpendicular directions. I'll briefly summarize the relevant properties in the next few paragraphs.

Looking down the photon's path, we can imagine placing a clock or a compass so that the photon goes through the disk's center. Let's use a compass here, as in Figure 4. We can set the polarizer so that the photon leaves vibrating in either the north–south plane (which I'll write as $|\uparrow\rangle$) or the perpendicular east–west plane (which I'll write as $|\rightarrow\rangle$).

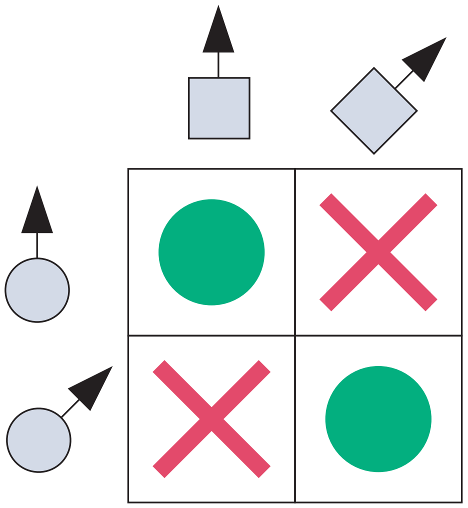
We can measure the photon with a *detector*, which is



5 When Bob measures a photon, his detector needs to be in the same orientation as Alice's polarizer. In this case, Alice has set her polarizer to setting *S*. (a) Bob sets his detector to *S* and correctly measures the incoming photon. (b) Bob sets his detector to *D*, and his measurement is random.

6 How Alice and Bob have agreed to match up photon polarizations with binary bits.

Basis	Value	Encoding
S	0	$\vert\rightarrow\rangle$
S	1	$\vert\uparrow\rangle$
D	0	$\vert\nearrow\rangle$
D	1	$\vert\nwarrow\rangle$



7 A summary of the results of the different possibilities for Alice's polarizer (circles) and Bob's detector (squares). The green circles indicate when Bob's measurement will be accurate. The red X's indicate when Bob's measurement will be random.

oriented as well. Figure 5a shows this. If we orient the polarizer in the north–south plane, then it can distinguish between north–south photons and those polarized east–west.

Now suppose that we rotate the polarizer 45 degrees, so that the photons are coming out in the northeast–southwest plane ($|\nearrow\rangle$) or the northwest–southeast plane ($|\nwarrow\rangle$). If we leave the detector in the north–south orientation, as in Figure 5b, then it can't disambiguate between these two cases. The detector will report about 50 percent of the photons as being in state $|\nearrow\rangle$ and about 50 percent of them in state $|\nwarrow\rangle$. The outcome will be completely random on a photon-by-photon basis.

If we now rotate the detector 45 degrees, then we can perfectly measure the two different states because we've aligned the detector in their directions. But if we send in a north–south photon, it'll now be randomly classified as one direction or the other.

The conventional way for Alice to encode her bits is shown in Figure 6.

Figure 7 summarizes the situation. If the detector is set randomly for each incoming photon, half the time it will correctly measure the photon's state. The other half of the time it will record noise. The detector will output a value, of course, but the value will be random and tell us nothing about the incoming photon.

These observations are the tools for a basic QKD system.

One essential step along the way to creating practical systems for dense coding, teleportation, and QKD is to find a way to create and maintain stable PDR pairs. This means getting quantum particles to become entangled, and then staying that way. I haven't focused much on these practical questions in these columns, but I'd like to mention a couple of recent developments that have people feeling very optimistic.

In a paper published this year, Kielpinski et al. reported on their work at NIST on creating entangled systems of beryllium ions. They've put together entangled systems of four ions in the state $\sigma(|\uparrow\uparrow\uparrow\uparrow\rangle + |\downarrow\downarrow\downarrow\downarrow\rangle)$. This opens the way for practical two-qubit computational devices.

More recently, Julsgaard, Kozhekin, and Polzik have published surprising results on entangling two huge clouds of cesium gas, each containing about a trillium atoms. Because they were entangled, large-scale changes in one cloud were mirrored in the other. The cesium clouds were held in place by magnetic fields inside vessels lined with paraffin wax. Entanglement is a fragile state, and left on their own it's not atypical for particles to disentangle within a million-billionth of a second. But because these clouds were so large, the system remained entangled for half a millisecond, which is an eternity in the quantum world. The two capsules were separated by several millimeters, which again is an astonishing distance in the quantum realm. Even more amazingly this all happened at room temperature!

Although these recent results in entanglement are causing a lot of excitement, they don't quite make for a complete, practical, and economic system for building and maintaining entangled systems of quantum particles. They give us plenty of hope though that such advances aren't far away.

With this optimism in place, let's take a closer look at how to use entangled particles to share secret messages in confidence.

Sending a quantum key

First I'll describe how quantum-key distribution works, and then I'll show why it's secure.

Alice's polarizer has two settings: *S* and *D*. If she sets the polarizer to *S*, then photons emerge in only the two states $|\rightarrow\rangle$ and $|\uparrow\rangle$. As you might expect, if Alice sets the

1.	0	0	1	0	1	1	0	1	0	0	1	1	0	1
2.	*S*	*S*	*D*	*S*	*D*	*D*	*D*	*S*	*D*	*D*	*S*	*S*	*D*	*S*
3.	→	→	↖	→	↖	↖	↗	↑	↗	↗	↑	↑	↗	↑
4.	*D*	*S*	*S*	*S*	*D*	*D*	*S*	*D*	*S*	*D*	*S*	*D*	*D*	*D*
5.	↖	→	→	→	↖	↖	↑	↖	→	↗	↑	↖	↗	↗
6.	×	•	×	•	•	•	×	×	×	•	•	×	•	×
7.		→		→	↖	↖				↗	↑		↗	
8.		0		0	1	1				0	1		0	

8 **QKD. 1. Alice's key. 2. Alice's polarizer settings. 3. The photons Alice sends. 4. Bob's detector settings. 5. Bob's measured photons. 6. Alice's report that tells Bob when he guessed wrong. × means an error, • means correct. 7. The photons Bob measured correctly. 8. The key Bob gets combining line 7 with line 4.**

polarizer to *D*, the photons will emerge in the two states |↗⟩ and |↖⟩.

Alice and Bob have agreed how to match up photon polarizations with bit values according to the table in Figure 6.

Remember at this point that Alice and Bob are only trying to transmit their key, not the message. Once the key has been exchanged and is secure, they can send the message publicly.

Refer to Figure 8 for an illustration of the following process; I'll refer to each row of this figure as we work through the process. Alice starts by using a random-number generator to create a binary key, (see line 1). The goal is to transmit this to Bob securely.

For each bit in the key, Alice randomly chooses either the *S* or *D* orientation for her polarizer and writes down her choice. This is line 2. Now Alice sends the key to Bob bit by bit, setting the polarizer to the appropriate setting and encoding her bits using the correspondence in Figure 8. This results in the photon string in line 3. Alice sends this string of photons to Bob over any public channel.

Now we turn our attention to Bob. He's got a detector with the same two settings, *S* and *D*. For each incoming photon, Bob randomly chooses an orientation, and positions the detector accordingly. Bob's choices in this example are in line 4. If it's more convenient, Bob can make his choices ahead of time and just consult the table as the photons come in.

Bob measures each photon using the current settings of his detector and writes down the measurement, resulting in line 5. Note that when Bob happens to guess right and has his detector oriented in the same way as Alice's polarizer, he correctly records the photon's state. But when Bob guesses wrong, he gets a random result.

When Alice has transmitted all the bits, Bob publicly sends Alice his detector settings from line 4. Note that he doesn't send his measured bits but just the settings that he chose to use. This information may be sent publicly. Alice then compares this to her private list (line 2) and sends back another public transmission telling Bob where he guessed incorrectly. Line 6 shows this public transmission.

Now Bob and Alice discard the bits where Bob guessed wrong (and thus got random measurements), as line 7 shows. The bits that were correctly guessed become the secret, one-time pad for Alice's upcoming message. Line 8 shows the string of bits that form the key; incorrectly guessed bits are just skipped.

Since Alice and Bob are both randomly picking their respective orientations, they expect to find that Bob guessed right about 50 percent of the time over the long run. So this process, while not stunningly efficient, isn't terribly inefficient either. If Alice wants to send an n-bit key, she only needs to send roughly $2n$ photons.

When Alice has finished sending the key to Bob, she can follow it up with a short test message, which she encrypts using the new key that she and Bob have just agreed on. She can send this message over public channels. Bob decodes it using the key, and then sends the resulting decrypted message back to Alice, again over a public channel if he wants.

Why would Alice and Bob do such a thing? It lets them confirm that their communications link is working, but there's an even better reason: this lets them determine if anyone was listening in when they exchanged their key. If the message that Bob sends back to Alice matches her original transmission, then their key is secure and they can start to trade secrets. But if anyone was listening in when they exchanged the key, about 25 percent of the bits that Bob sends back will be wrong, and this will clearly identify that someone was eavesdropping.

To me, this is one of the most beautiful results in quantum computing. Let's see where it comes from.

Catching eavesdroppers

Let's suppose that while Alice is sending her photons to Bob—that is, sending the information on line 3 of Figure 8—an interpoloper named Eve is eavesdropping.

Let's think carefully about what everyone wants in this process. Alice and Bob want to exchange their key accurately and securely. That is, Bob has to get the correct bits, and they both have to know that nobody else has them.

Eve's options are a bit broader. She might simply want to stop Alice and Bob from communicating at all, by blocking their transmission channel. That's effective but not very subtle, and it gives away her presence. Eve's real goal would be to catch a copy of the key in secret. Then Alice and Bob would continue exchanging messages thinking that they were secure, while Eve decrypts each one and acts on her ill-gotten knowledge. Therefore, Eve's goal is to get a copy of the key without revealing her presence. In terms of Figure 8, she needs to catch the photon stream in line 3 that Alice is sending to Bob.

Eve's fondest wish would probably be to capture the photons, make copies of them, and then forward one copy to Bob while keeping a copy for herself. Her best approach would be to make copies of the photons without measuring them first, but that would require cloning, and we saw earlier that quantum cloning is impossible.

So Eve's best remaining bet would be to intercept each

9 How Eve's interception changes Bob's results. In each column, Alice is sending a 0 bit using the *S* setting on her polarizer. The lines have the same meanings as Figure 8, except for new lines, Q and R. Q shows Eve's detector settings, and R shows Eve's measured photons, which she sends to Bob. In column 1, Eve and Bob both guess correctly, and Bob gets the correct bit. In columns 2 and 4, Bob guesses incorrectly so his measurements are thrown away. In column 3, Bob guesses correctly but Eve guesses wrongly. Half the time Bob will measure the value that Alice sent, and half the time he'll get it wrong. The symbol ∅ indicates a noisy, or random, measurement.

1.	0	0	0	0
2.	S	S	S	S
3.	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$
Q.	S	S	D	D
R.	$\rightarrow$	$\rightarrow$	$\varnothing$	$\varnothing$
4.	S	D	S	D
5.	$\rightarrow$	$\varnothing$	$\varnothing$	$\varnothing$
6.	•	×	•	×
7.	$\rightarrow$		$\varnothing$	

10 QKD with eavesdropping. The lines have the same meanings as in Figure 9, with the addition of Line 9, which presents the results when Bob and Alice compare their bits. Notice that information can be corrupted either when Eve guesses wrong (on Line Q), or when Bob guesses wrong (on Line 4). In Line 8, I've chosen a random value for the bit in cases when Eve corrupted the data and sent a random bit. The symbol × means an error, • means correct. Generally, about half of the bits marked ∅ on Line 7 will result in errors on Line 8. The presence of errors in Line 9 reveals that Eve was listening in on the exchange.

1.	0	0	1	0	1	1	0	1	0	0	1	1	0	1
2.	S	S	D	S	D	D	D	S	D	D	S	S	D	S
3.	$\rightarrow$	$\rightarrow$	$\nwarrow$	$\rightarrow$	$\nwarrow$	$\nwarrow$	$\nearrow$	$\uparrow$	$\nearrow$	$\nearrow$	$\uparrow$	$\uparrow$	$\nearrow$	$\uparrow$
Q.	S	D	D	S	S	S	D	S	S	S	D	D	D	S
R.	$\rightarrow$	$\varnothing$	$\nwarrow$	$\rightarrow$	$\varnothing$	$\varnothing$	$\nearrow$	$\uparrow$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\nearrow$	$\uparrow$
4.	D	S	S	S	D	D	S	D	S	D	S	D	D	D
5.	$\varnothing$	$\varnothing$	$\varnothing$	$\rightarrow$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\nearrow$	$\varnothing$
6.	×	•	×	•	•	•	×	×	×	•	•	×	•	×
7.		$\varnothing$		$\rightarrow$	$\varnothing$	$\varnothing$				$\varnothing$	$\varnothing$		$\nearrow$	
8.		0		0	0	0				1	1		0	
9.		•		•	×	×				×	•		•	

photon as it comes by, measure it, create another photon in the same state, and send that on to Bob. In that way, she gets to read each photon and Bob is none the wiser.

Well, that almost works, but not quite. In fact, Eve will give herself away if she tries this. The problem is that Eve doesn't know how to measure the photons she's intercepted. Like Bob, she is unaware of the settings that Alice used on her polarizer. So, like Bob, Eve must guess at a setting for her detector, and again like Bob, she has a 50–50 chance of getting it right for any given photon. When Eve's guess at a detector setting matches Alice's polarizer setting, Eve measures the photon properly, but the other half of the time she gets noise.

So Eve just goes forward as best she can. After measuring the intercepted photon, she creates a copy in the same state and sends it on to Bob.

Let's look at what can happen to the first bit of the message. Figure 9 shows the possibilities.

In the first column, Alice sets her polarizer to *S*, and sends a photon in state $|\rightarrow\rangle$. Eve is lucky and guesses *S*, measures the photon as $|\rightarrow\rangle$, and sends it on to Bob. Bob is also lucky, guesses *S*, detects $|\rightarrow\rangle$, and Eve appears to have gotten away with it.

In the second column, Eve guesses correctly and sends along a $|\rightarrow\rangle$ photon to Bob, but he guesses *D* for his detector. We know this bit will get thrown out later when Bob tells Alice that he guessed *D*, so again Eve has gotten away with her eavesdropping.

In column 3, Eve guesses wrong and sets her detector to *D*. This means that her measurement is going to be randomly either $|\nearrow\rangle$ or $|\nwarrow\rangle$. I've indicated this with ∅, to indicate that any information about the original photon has been lost (I'm using the symbol ∅ to indicate a random value). Eve makes a copy of her received photon (in state $|\nearrow\rangle$ or $|\nwarrow\rangle$) and forwards it to Bob. Now in this case, Bob sets his detector to *S*. What does he measure? Remember that the photon he's receiving now is random. When Eve measured it, she projected it into one of the *D* states. So even though Bob's got the right setting on his detector, he's getting a photon that is randomly oriented in one of the *D* states. So half the time he will measure the photon as $|\rightarrow\rangle$, and half the time as $|\uparrow\rangle$. That's where Eve gets caught, as we'll see in a moment.

Finishing up in column 4, Bob here again guesses incorrectly in his detector setting, so it doesn't matter what Eve did.

Now when Eve isn't listening in on the conversation, every time Bob guesses Alice's orientation correctly, he will correctly decode Alice's bit. But when Eve is listening, then we've seen that about 25 percent of the time Bob will guess correctly, but still get the wrong bit. The error occurs when Eve guesses wrong and sends Bob noise (which she does half the time), and then Bob measures that noise, so he has only a 50–50 chance of measuring the photon in the same state that Alice intended.

Now we see the value of the test message I described earlier. Alice encrypts a piece of plaintext and sends it to Bob using the new key, using a public channel if she wants. Bob decrypts it and sends Alice back the resulting plaintext, again publicly if he so deires. If about 25 percent of Bob's bits are wrong, Eve has been caught.

Figure 10 shows one possible result of Eve's interloping on the signal of Figure 8.

There's no way for Eve to get around this problem; she's blocked from every direction. Quantum uncertainty means that she can't measure the photons correctly every time. Because measurement changes the photon's state, she can't measure it twice. Because she can't clone the incoming photons, she can't send an undisturbed stream to Bob.

The very fundamentals of quantum mechanics mean that although she can get away with capturing a few of the bits of the key, over the long haul her eavesdropping will be caught. It doesn't even have to be that long a haul. With a 25 percent error rate, she's going to be detected easily and quickly.

Once Alice and Bob know that Eve listened in, they're safe. They might be inconvenienced, but their secret won't be compromised because they know not to send it in the first place. They would probably choose to try again with a new key (perhaps using another communications channel) and hope Eve isn't listening in this time. Of course, Eve can listen in on every attempted key transmission and thereby prevent Alice and Bob from exhanging encrypted messages, but she'll never be able to listen in on the secrets they do send.

You might like thinking about other strategies that Eve could employ to make a copy of the key without giving herself away. So far, nobody has found a way for Eve to copy the key without revealing herself in some way.

Of course, any cryptographic system can be compromised in practice in ways that have nothing to do with the value of the theory the system is based on. For example, a big break in cracking the Enigma system used in World War II came from its operators failing to initialize the machine properly. At least in theory, quantum-key distribution so far seems to offer an ironclad way of creating private one-time keys. The only way to stop Alice and Bob from exchanging secrets is to always eavesdrop or prevent them from communicating at all.

Quantum algorithms

Two basic papers are widely considered to be the seminal works in the field of quantum computing. In 1994, Peter Shor invented an algorithm for factoring large numbers. This algorithm catapulted quantum computing into the front pages of newspapers and magazines around the world, and sparked a lot of interest among theoretical and practical physicists and computer scientists.

As we've discussed, cryptography is an important technique in all sorts of communication and information storage methods. Of the many cryptographic systems currently in use on computers, the RSA system is one of the most popular. This technique is based on the observation that factoring prime numbers is apparently a hard problem. Suppose I pick two large, relatively prime numbers M and N and compute a new number $P = MN$. I can make P so large that it has hundreds or even thousands of decimal or binary digits. Now suppose that my encryption scheme is based on the numbers M and N, but I only reveal P. It's currently prohibitively time-consuming to start with a huge number like P and find its prime factors M and N. Mathematicians and computer scientists have spent a lot of time and energy on this problem, and it remains difficult. Factoring prime numbers isn't probably hard in theoretical sense; there could be a fast and simple algorithm lurking right around the corner that nobody has found yet. But because researchers haven't found such an algorithm, RSA has been widely adopted for keeping secrets.

Shor's algorithm dropped a bombshell in the cryptographic community, because it showed how a quantum computer could factor even enormous prime numbers without taking forever. Should a quantum computer of enough power be built, the secrets encoded by RSA since it was first developed might be easily read by everyone.

The second big development came in 1996, when Lov Grover presented an algorithm for finding a particular

Further Reading

As in the previous columns in this series, I used many papers publicly available on the Los Alamos Physics Preprint Archive at Los Alamos National Labs, located at http://xxx.lanl.gov/abs/quant-ph. In particular, see the paper by Eleanor Rieffel and Wolfgang Polak for the proof of the impossibility of quantum cloning.[1]

The quantum-key distribution scheme I presented is commonly called BB84 after its inventors, Bennett and Brassard, and its publication year. It was republished in 1985.[2]

The recent breakthrough in macroscopic entanglement by Julsgaard, Kozhekin, and Polzik was published in the September 27, 2001 issue of *Nature*. You can find an online preprint of the article in the LANL archives.[3]

Entangled states of four trapped ions have been produced in the laboratory. Details appear in the article by Sackett et al. in *Nature*.[4] You can read an earlier version of this article at the LANL archives.[5]

You can find a nice overview of quantum cryptography in "From Quantum Cheating to Quantum Security" by Daniel Gottesman and Hoi-Kwong Lo. It's available at http://www.physicstoday.org/pt/vol-53/iss-11/p22.html.

Shor's algorithm for factoring large numbers appeared in 1997.[6] The text is also available online from Peter Shor's Web page at http://www.research.att.com/~shor/papers/.

Grover's searching algorithm is available in several forms. One summary is on the LANL archives.[7] A published version appears in the *Physics Review Letters*.[8]

To test out quantum programs, you'll either need to build a quantum computer or else a quantum-computer emulator. Unless you've got a physics lab handy, I recommend the latter. It's easier to get going and cheaper, too! Happily, there are some available for free. Hans De Raedt, Anthony H. Hams, Kristel Michielsen, and Koen De Raedt describe the QCE program for Windows.[9]

References

1. E. Rieffel and W. Polak, *An Introduction To Quantum Computing for Non-Physicists*, LANL 9809016, 1998.
2. C.H. Bennett and G. Brassard, "Quantum Public Key Distribution," *IBM Technical Disclosure Bulletin*, vol. 28, 1985, pp. 3153-3163.
3. B. Julsgaard, A. Kozhekin, and E.S. Polzik, "Experimental Long-Lived Entanglement of Two Macroscopic Objects," LANL 0106057, 2001.
4. C.A. Sackett et al., "Experimental Entanglement of Four Particles," *Nature*, vol. 404, 2000, pp. 256-259.
5. D. Kielpinski et al., "Recent Results in Trapped-Ion Quantum Computing," LANL 0102086, 2001.
6. P.W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms On A Quantum Computer," *SIAM J. Computing*, vol. 26, no. 2, Apr. 1997, pp. 1484-1509.
7. L.K. Grover, *A Framework for Fast Quantum Mechanical Algorithms*, LANL 9711043, 1997.
8. L.K. Grover, "Quantum Mechanics Helps In Searching For A Needle In A Haystack," *Physical Review Letters*, vol. 79, no.2, July 1997, pp. 325-328.
9. H. De Raedt et al., *Quantum Computer Emulator*, LANL 9911041, 2000.

item in a database of N items in only $O\sqrt{N}$ steps; roughly that means it will take only $k\sqrt{N}$ steps for some value of k. This is important because many important problems can be thought of as searching problems, or can be mapped into them. I used a variant of this algorithm in part 1 of this series to find the minimum-energy solution to a radiosity simulation.

I won't present these algorithms here because they are nontrivial. I wanted to mention them so you'd be familar with these famous names and techniques. The "Further Reading" sidebar points to some articles that contain both high-level and detailed descriptions of these algorithms and their implications.

Graphics algorithms

In part 1 of this series, I described an algorithm for quantum radiosity. By now you've probably thought of a dozen more graphics algorithms that could be made faster or even practical for the first time on a quantum computer.

The first place to go hunting for cool quantum computing applications is anything involving searching for a minimum or maximum because Grover's algorithm can speed this up significantly. The Z-buffer is a prime candidate for this kind of technique. With a quantum computer we wouldn't have to draw in all of our micropolygons as they come by, discovering the nearest one simply because it's the only one left after all the scan-conversion has finished. Rather, we can attach one quantum computer to each pixel and throw all of our micropolygons at each one. Then, we use Grover's algorithm to find the micropolygon with the smallest Z value, shade it, and fill in the pixel.

Ray tracing is another brute-force searching algorithm that would benefit from this sort of speedup. For simplicity, suppose we had a picture made up entirely of spheres (ray-tracing people just love spheres). Finding a ray-sphere intersection involves solving for the roots of a quadratic polynomial and then typically sorting for the smallest nonnegative root (see any book on ray tracing for the details of this calculation). The polynomial we want to solve is $at^2 + bt + c$, and we're looking for the values of t, which are given by the formula we learned in high-school algebra: $t=(-b\pm\sqrt{b^2-4ac})/2a$. Each sphere has its own values a, b, and c and results in its own pair of t values. (If there's no intersection, we still get values for t, but they're imaginary.) The value of t tells us how far we need to travel along the ray from its starting point until we reach the sphere.

To find the nearest intersection of a given ray with all the spheres, load up three quantum registers with a superposition of all the values of a, b, and c for all the spheres in the scene. Now compute the two roots of t with these superimposed registers. Just like that, in one calculation, you have the t values for every sphere in the scene, all at once! Now you just need to use a clever technique like Grover's algorithm to bump up the probability value associated with the state that has the smallest real nonnegative value of t. Measure the t register and out pops the number.

In practice, you'd probably want to attach an object ID to the t register, so you get back not just the smallest value of the ray parameter but also the identifier for the object that was hit. This will be necessary to look up information like the surface normal, materials and textures, and so on.

There are few things that we can count on from developing technologies, but change is one of them. Quantum computing is going to create a lot of change in the world of computing, including computer graphics. These articles have only described the beginnings of a field that is less than a decade old. When engineeers and physicists work out ways to build large and economical quantum computers, hang onto your hats because everything we know about computers is going to be up for grabs. ■

Acknowledgments

Thanks again to Kirk Olynyk and Don Mitchell for their help with this series of columns.

Readers may contact Glassner by email at andrew_glassner@yahoo.com.