BLACK ROSE TECHNOLOGY

QUANTUM COMPUTING AND CRYPTOGRAPHY

BRT-9-W-23-0013

DELIVERED AT DEFCON 9.0, JULY 2001

Title	Quantum Computing and Cryptography
Document Number	BRT-9-W-23-0013
Revision	0
Page Count	39
Date Created	2023-07-07
Last Modification	2023-08-06
Author	David Gessel
Checked by	DJG

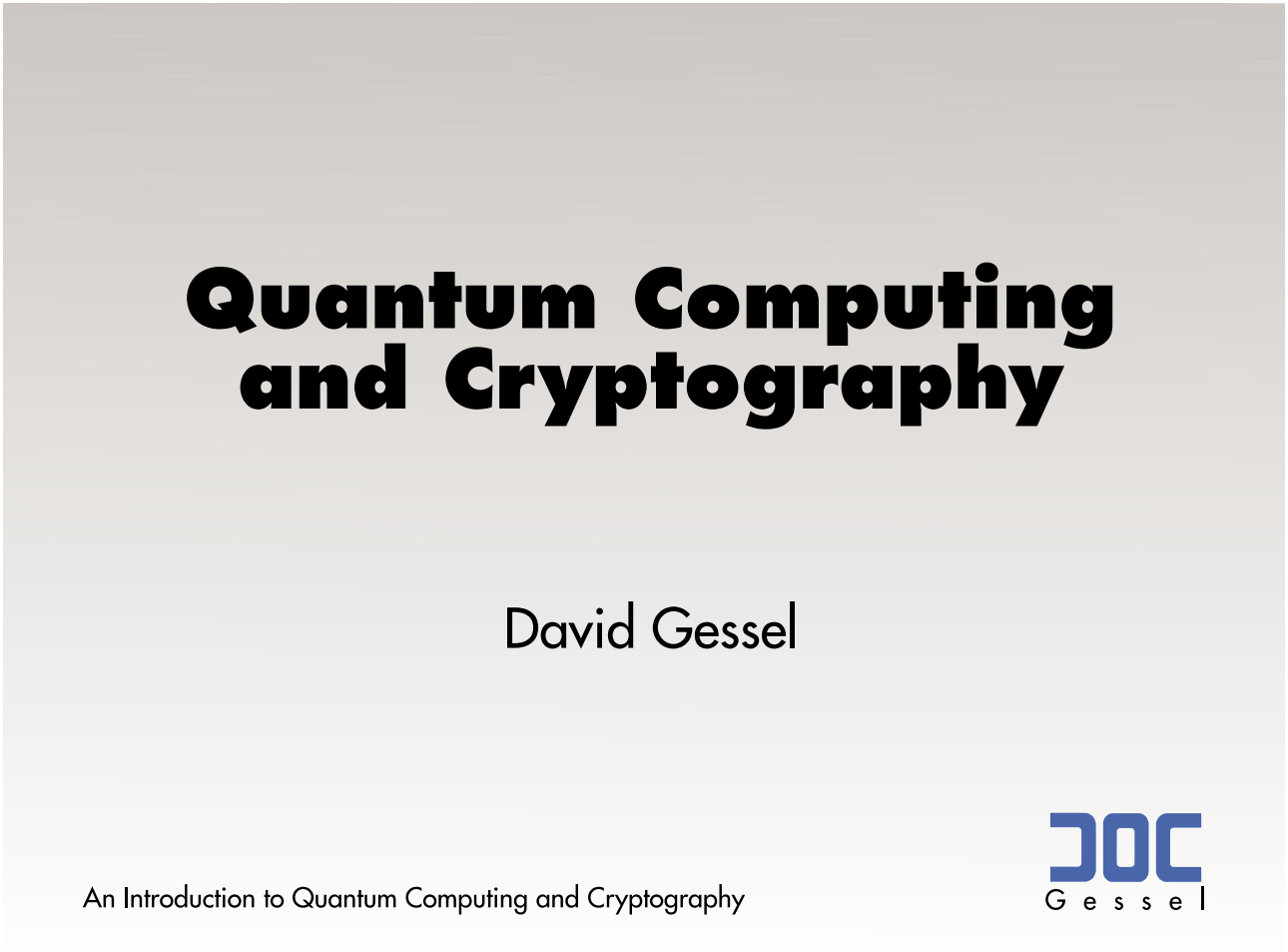
QUANTUM COMPUTING AND CRYPTOGRAPHY

DEFCON 9.0, 15 JULY 2001, 10:00

ALEXIS PARK RESORT, LAS VEGAS, NEVADA

EDITED TRANSCRIPT WITH SLIDES

This is a transcript of the original talk extracted with speech-to-text by whisper and then edited for clarity and to clean up some quirks. It is therefore in a conversational linguistic style.



Today's [15 July 2001] talk is going to be about quantum cryptography and quantum computing. It's an introductory talk, everything I'm going to talk about requires very minimal math. As one can imagine, quantum computing and cryptography is a large subject, and we're going to be moving quickly through a lot of possibly new concepts, especially if one hasn't had much exposure to quantum mechanics in the past, but hopefully, if all goes well, by the time we're done we should have a sense of the basics, including:

1.0 TALK STRUCTURE

1.0 Introduction

2.0 Classical computing, basic definition
3.0 Basic principles of Quantum Mechanics
4.0 Basic principles of Quantum Computing
5.0 Applications: Cryptography, Cryptoanlysis
6.0 Practical Implementations
7.0 Conclusion

An Introduction to Quantum Computing and Cryptography

DOC Gessel

The basic definitions of classical computing and how it relates to quantum computing and how it might be different, the basic principles of quantum mechanics and how they relate to the basic principles of quantum computing, some applications of quantum computers, and a few practical implementations. Then I'll talk a little bit very briefly about where things are going. So let's get right into it.

2.0 CLASSICAL COMPUTING & BASIC DEFINITIONS

2.0 Classical Computing

2.1 Turing Machines
2.2 Dimensions: Clock, Complexity, Parallel
2.3 P, nP, Hard Problems, and Intractability

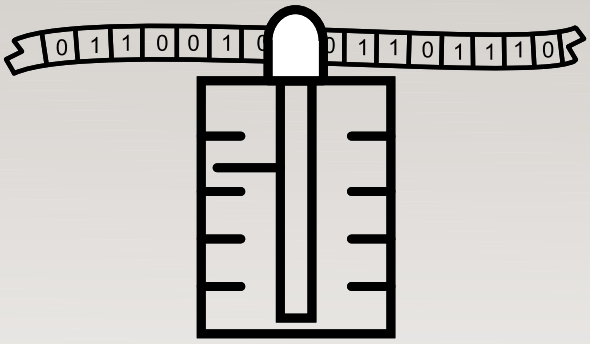
An Introduction to Quantum Computing and Cryptography

Gessel

In classical computing, the first thing to know about is the definition of Turing machines. We'll also cover, briefly, complexity and some definitions of how really hard problems are different from just normal difficult problems.

2.1 TURING MACHINES

2.1 Turing Machines



Reads one bit at a time from the tape, depending on the internal state, writes a new bit on the tape
Church's Thesis: Any computable function can be computed on a Turing machine. (Approx 1930)

An Introduction to Quantum Computing and Cryptography

DOC Gessel

A Turing machine reads one symbol at a time, then changes its state internally based on that one symbol and the internal state at reading and then outputs a symbol and reads another symbol, either forward or backward. Theoretically, it could have an input tape and an output tape, and it wouldn't be any different it's just a little more elegant to draw it this way.

"...a certain type of discrete machine was described. It had an infinite sensory capacity obtained in the form of an infinite tape marked out into squares on each of which a symbol could be printed. At any moment there is one symbol in the machine; it is called the scanned symbol. The machine can alter the scanned symbol and its behavior is in part determined by that symbol, but the symbols on the tape elsewhere do not affect the behavior of the machine. However, the tape can be moved back and forth through the machine, this being one of the elementary operations of the machine. Any symbol on the tape may therefore eventually have an innings."

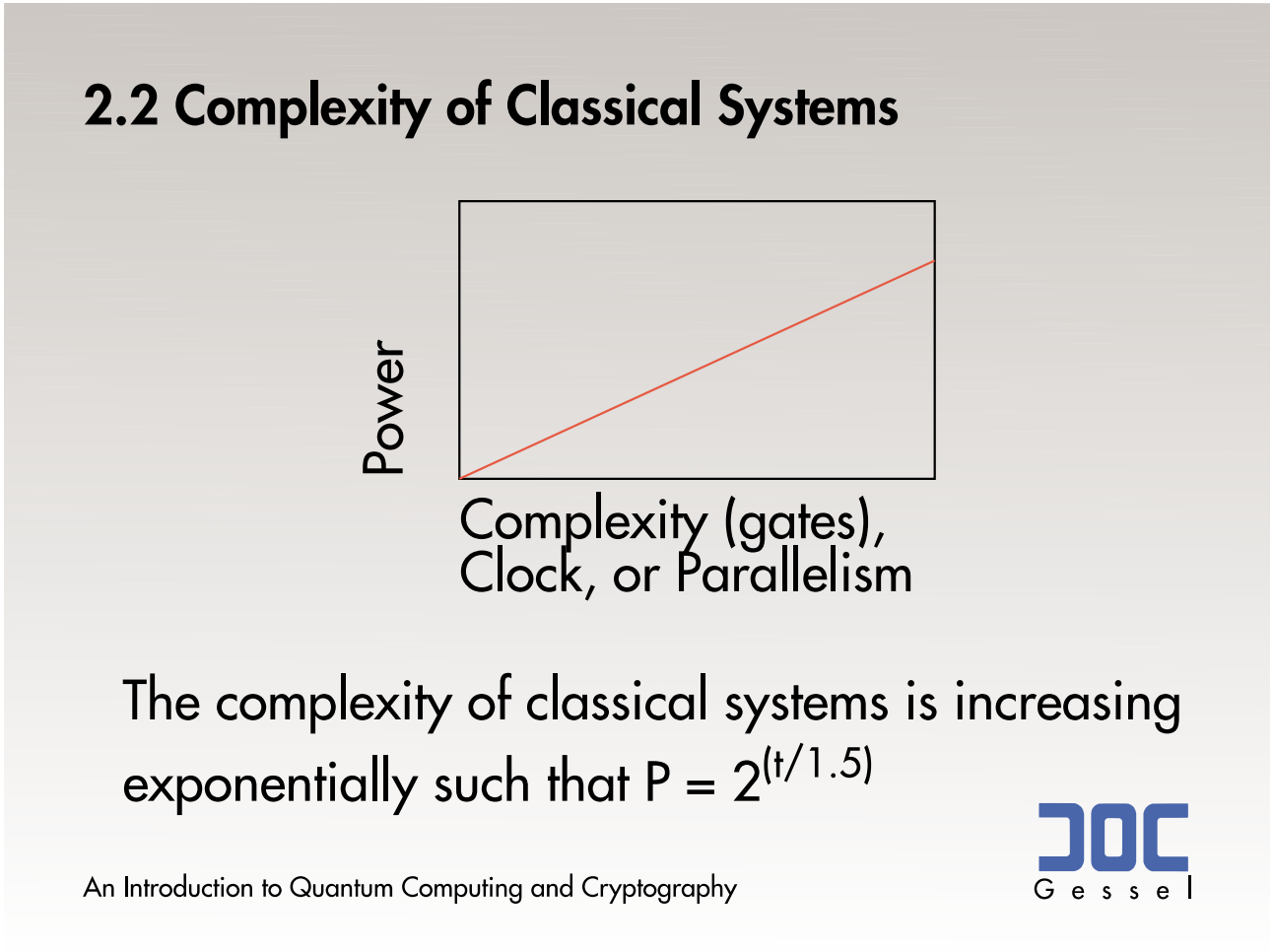
— A.M. Turing, Jul 1948

What's interesting about this machine, simple as it might seem, is encapsulated in Church's thesis formulated in about 1930: any computable function can be completed on a Turing machine.

"No computational procedure will be considered as an algorithm unless it can be represented as a Turing Machine"

Anything that we can compute can be done with these simple steps.

2.2 DIMENSIONS: CLOCK COMPLEXITY, PARALLELISM



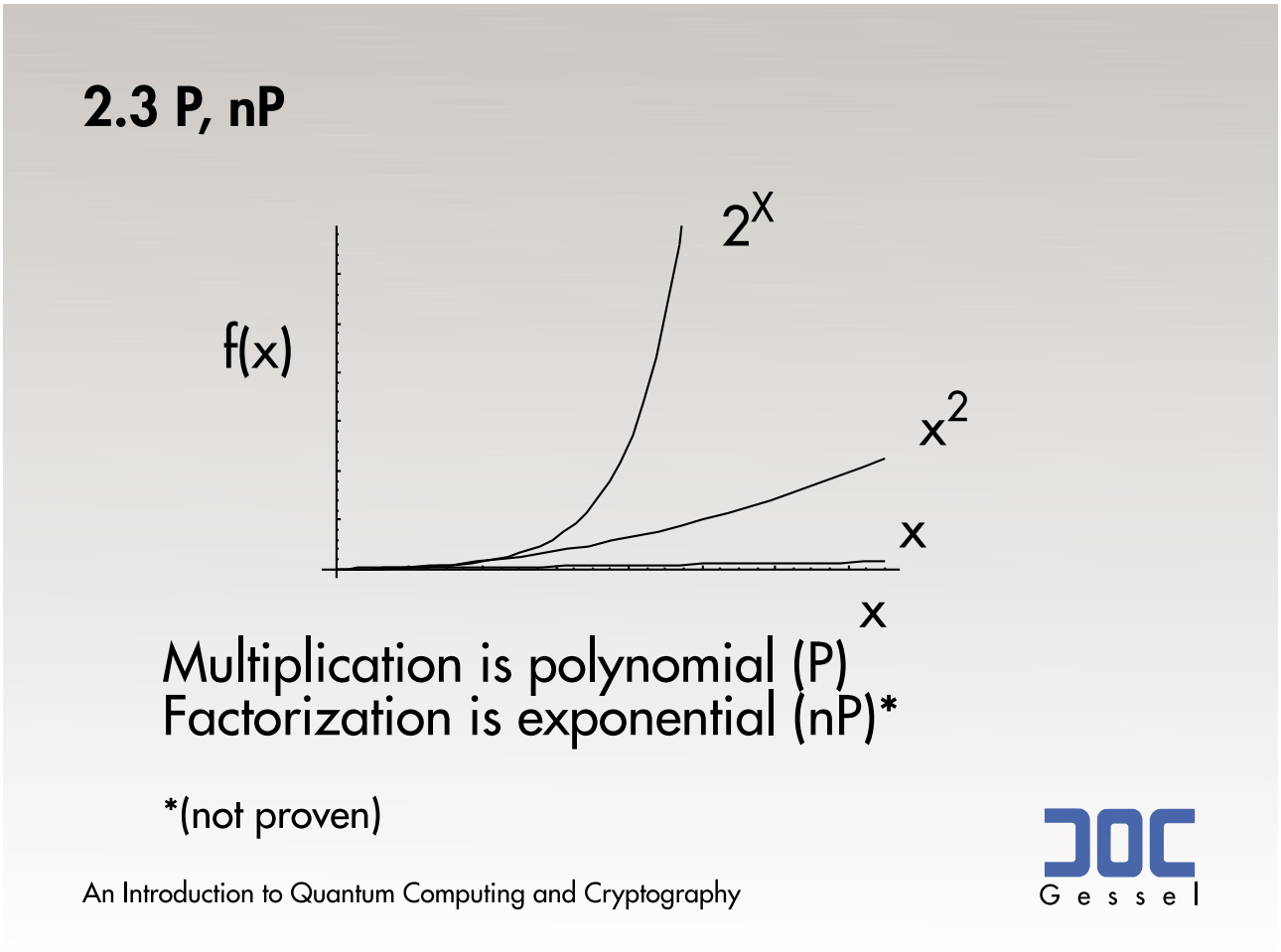
Another interesting note that applies to quantum computing is Moore's law, which can be interpreted as the complexity of computers, of classical computers, has been increasing exponentially over time. That is, the number of gates in a classical computer has increased exponentially since about 1965 and continues to do so today, at least with some slight variations in the definition to accommodate changes in technology and having overcome many theoretical bounding limits. One of the driving forces behind discovering quantum computing was the realization that, despite circumventing many theoretical barriers such as limits of optical resolution, sooner or later we're going to be dealing with on-chip features the size of single atoms and at that point quantum physics dominates, and classical systems run out the limit of areal density improvements.

So people began looking at quantum computing as an answer, initially, to the problem of what would happen when we reached the limiting size of features, or the quantum structural limit on classical computers.

Until now, the power of classical systems, because of the technological advances, has been increasing at a rate 2 to the power of years divided by 1.5, which means the doubling period for computing power is every 18 months, and that's Gordon Moore's famous law: $2^{(t/1.5)}$ where time is measured in years. The power of a classical system as a function of complexity is roughly linear, which is to say that we don't get a faster computer without more complexity. Our computer is twice as powerful if we have twice as many gates (or, in some applications, twice as much memory), at least to a first approximation.

We may consider that computational power increases roughly linearly with the number of gates. There have been improvements that have changed this rule a little bit, but basically it increases linearly with complexity and complexity increases exponentially with time. That, we'll soon see, is a significant statement and a significant difference between classical computers and quantum systems.

2.3 P, NP, HARD PROBLEMS, AND INTRACTABILITY



If we look at the problems that are applied to classical systems, they fall into two basic mathematical classes (possibly, though that hasn't yet been proven they're fundamentally different).

The first are problems that are computable in quadratic time, called P problems, for polynomial time; problems that fall into this class take either twice as much time to compute or a power of, say, x^2 as much time as the complexity of the problem, compute load may be raised to a power of the complexity (or size) of the problem, but not as a constant to the power of the complexity.

A good example of this class of problem is multiplication: it generally takes about twice as long to multiply two twice larger numbers as two half smaller numbers: t clocks to multiply X×Y, 2t clocks to multiply 2X×Y. The complexity of multiplication is a linear function with the size of the values being multiplied, which means that, in general although not necessarily in practice, P problems, problems that can be solved in polynomial time, are considered tractable problems even at very large scale. These are categories of problems that a classical computer can be applied to meaningfully, and we can expect to get a solution in some reasonable, human-scale time period.

If we're doing something very complicated with a classical computer, for example number theory or simulating some complex mechanical system, which may be defined as a theoretically tractable problem the total number of computations may well be so large that we won't want to wait for it to complete, but at least in theory, if we double the complexity of the problem, it will only take twice as long, or complexity to a fixed power as long; it's a polynomial problem, and it's considered tractable.

The top trace shows what happens when we have problems that are not polynomial, that is, nP problems. It may be true that nP problems are actually P problems, it hasn't been proven in a mathematical sense that they're truly different, but this

is assumed to be so and people have worked really hard to find an answer as to whether nP problems are different from P problems without success. nP problems are problems that take two to the complexity of the problem longer, or some other constant to the power of the complexity longer to solve, 2^x, not x^2. That is, the time to solve an nP problem rises very, very quickly with the complexity of the problem.

A canonical examples of a problem that are well known to be believed nP problems is factoring of large numbers. The reason that's important is that the difference between the ease of multiplying two large primes, which rises linearly with the size of the primes, and the difficulty of factoring a large semiprime (the product of two prime numbers) is the basis of cryptography.

We've established it is very easy to multiply two large numbers together, and it doesn't get much more difficult if the numbers get larger, but to find the common factors of a very large number takes on the order of 2^x operations where X is the size of the number.

More explicitly, it takes x time where x is the total number of digits to multiply two large numbers together, depending on the function used (or possibly x^2 or x^3 or some polynomial combination) but it takes 2^x operations to factor the result.

Very crudely, if multiplying two 16-bit numbers together takes 32 time, factoring the 32 bit result takes 4,294,967,296 time.

And that's how cryptography works, or at least public key cryptography. The basic premise is that we can make a function that's very easy to solve in one direction, but nobody can, in practice, solve it in the other direction because very quickly the problems becomes intractable. A problem that can be solved one way in human time but in reverse can't be solved in human (or even geologic) time is something we can build a secure cryptographic system around.

But a Turing machine can solve any problem, even an intractable problem, right? True, we know that it can solve all problems, but when the amount of time it takes to find a solution, even when we take into account Moore's law and the exponential increase of the power of computers, would be longer than the predicted lifetime of the universe we can comfortably say that's an intractable problem.

And factoring very large numbers quickly falls into that category: factoring thousand-digit numbers on standard (2001 vintage) computers, even taking into account Moore's law would take longer than the projected lifetime of the universe. Which is why people today believe that if they use large keys, 1024, 2048-bit keys, for their public key encryption, that they're probably going to be safe, at least for their lifetime.

It is possible that there's a shortcut to factoring, in fact, there's a number sieve theory that's cut the time down significantly, but it hasn't changed the problem from an nP problem to a P problem. It's still an nP problem. Furthermore, it's just a little bit less intractable for cryptographically short keys, which are now, thanks, considered insecure.

Using these methods, 128 & 129-bit numbers have been factored in a reasonable 42 days using a teraflop or so of processing power. So it is tractable to do for smaller keys once thought secure thanks to some advances in mathematics, but it's still believed to be completely intractable for large keys, roughly doubling in compute time for each additional bit of key length.

To introduce a more philosophical than practical idea which is derived from Turing's thesis, consider that mathematics is the derived language of the universe, that is, the language of math is universal, not necessarily because everybody thinks about numbers in the same way or all cultures do, but if we notice that, generally, although some cultures didn't invent zeros and some did, basically all cultures have figured out addition and subtraction.

And those two basic operations are all it takes to make a Turing machine.

"*Every finitely realizable physical system can be perfectly simulated by a universal model computing machine operating by finite means.*"

— *David Deutsch, July 1984*

3.0 BASIC PRINCIPLES OF QUANTUM MECHANICS

3.0 The Basics of Quantum Mechanics

3.1 Uncertainty and Heizenberg
3.2 Spin, Polarization
3.3 Two Slit Experiment
3.4 Quantum Interference
3.5 Spooky Action at a Distance

An Introduction to Quantum Computing and Cryptography

DOC Gessel

There's a derived language of the universe, which is mathematics, and what's been amazing about that derived language is that quantum mechanics was derived, the theories of quantum mechanics were derived, from the mathematics of the problems considered and later validated experimentally rather than by a contemplation of reality directly.

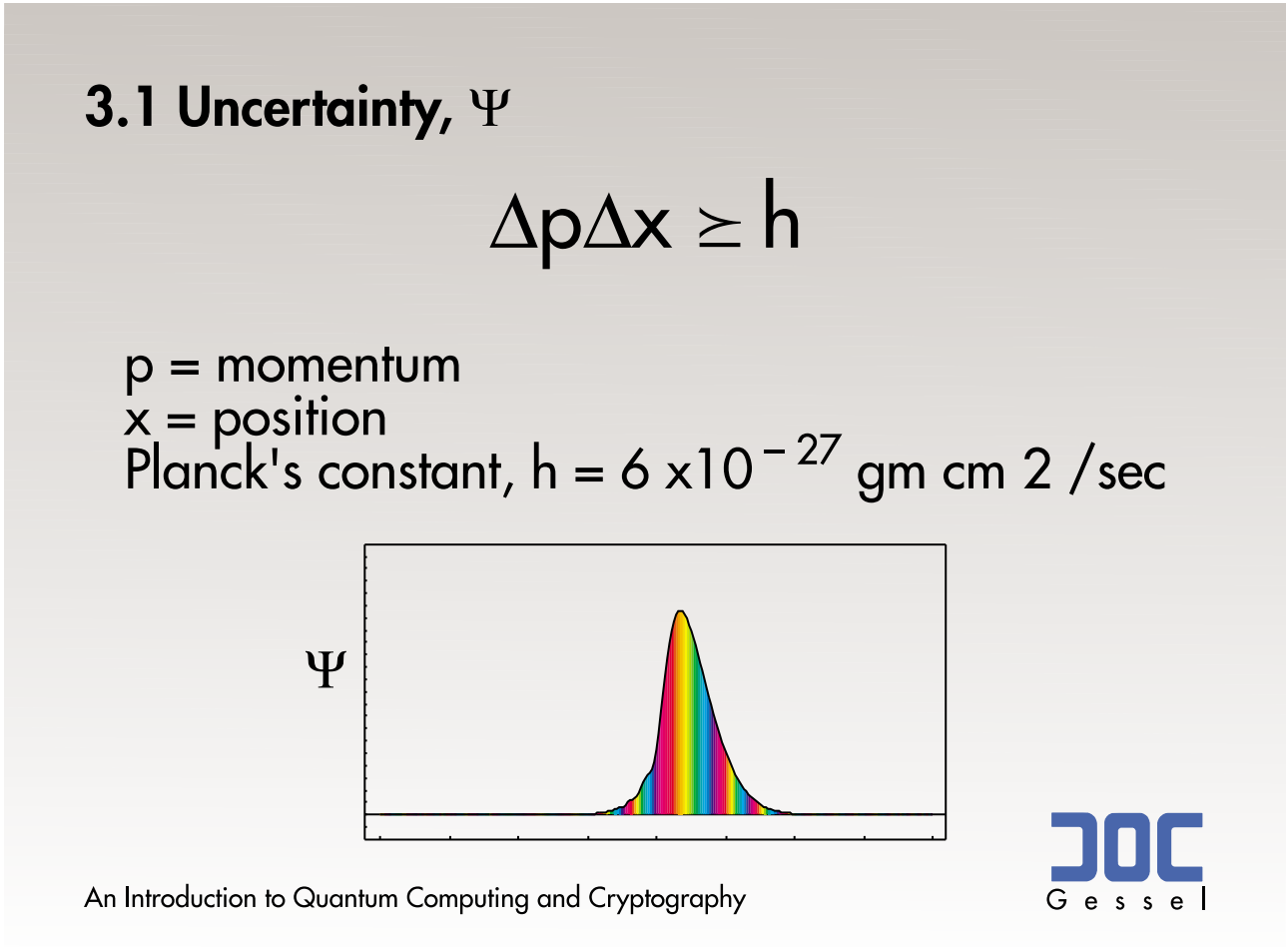
The rules of quantum mechanics were not derived from anomalies in direct observables, but rather from anomalies in the mathematics and observations of quantum mechanical effects that proved quantum mechanical theory didn't happen until later, validating the predictions of the mathematics.

The basic definition of the universe, which turns out to probably be nominally correct, is quantum mechanics as it stands today or will stand in the near future, and was derived mathematically from the abstract rather than from observations of the existing universe, as if by reading the notes ahead of the production.

What's interesting about when we do this derivation is that we come up with some very strange predictions, things that don't jive at all with our experience of everyday life, in fact, they don't appear in our experience of everyday life at all, ever, and therefore it was reasonable to be highly suspect if not dismissive, but the strange phenomenon that can be tested, have been tested, and reality rather bizarrely lined up with the mathematical predictions.

And some of those quantum predictions are pretty spooky, indeed. 👻

3.1 UNCERTAINTY AND HEISENBERG



One of the basic foundations of quantum mechanics is Heisenberg's Uncertainty Principle, which we may be passingly familiar with for the famous cat memes, but superficially it says we can't know the position and the momentum of a particle or any entity exactly at the same time; we can't know both values exactly at exactly the same time.

We could know one and then the other with a time delay between them; we could know one or the other at the same time; but we can't exactly know both of them at the same time.

That's a very unusual statement. It's something that doesn't jibe with anything that we see in the everyday world. It's a completely new concept alien to classical physics. Furthermore, it's a profound statement because what it means is there's a fundamental limit, a quantum mechanical limit, to where we can find things. But there's another thing that it also means, which is something that really bugged Einstein, which is that at a basic level it appears that the universe is stoichiometric, that is, things happen by chance or appear to happen by chance. And Einstein famously said, "*God doesn't play dice*," and he wanted to try to find some way around it but never did.

There are arguments that were happening within the Copenhagen School of Physics around 1925–1927, the philosophical birthplace of quantum mechanics, between a bunch of people there who worked through a lot of problems and came up with a lot of seeming contradictions. Things that Einstein felt, and I'll get to one of the more famous and useful ones soon, things that Einstein felt really indicated that there was something lacking in quantum theory, that it must be incomplete.

But all of these things have been found, if not necessarily easily comprehensible definitions, the basic concepts have been proven out in every experiment to the maximum precision that's possible so far.

Quantum mechanics introduces a weird concept that at the most basic level a particle or object does not have a defined instantaneous location *and* momentum. From that bit of strangeness, and from some other weird things, we derive the wave function, Schrödinger's equation, which is the colorful graph in the slide. Those are not my colors. I assume that they describe phase transition and a moving particle, but that's not relevant to our discussion. We'll just enjoy the colors.

$$H(t)|\psi(t)\rangle = i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle$$

What Schrödinger's equation describes is the location of a particle and what we should notice from the graph is that's an asymptotically decreasing function, a bell curve, roughly, which doesn't go to zero anywhere.

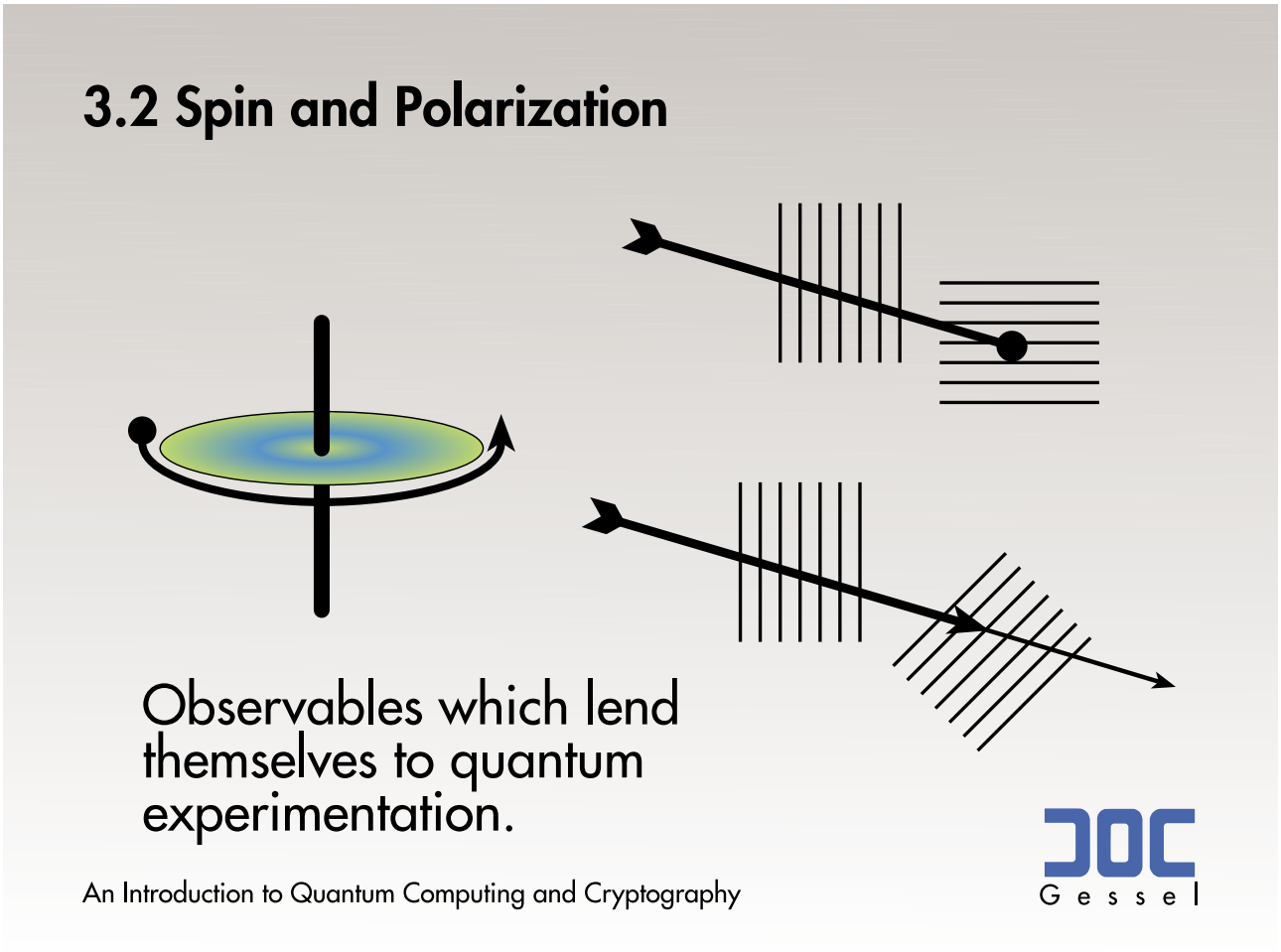
We can calculate the probability of finding a particle at any distance from the classical location of that particle at any time, any distance anywhere and anytime in the universe. A particle is not in any one place: it's everywhere, and possibly everywhen, at least until it has been measured. It is, however, mostly where and when we think we're going to find it. Mostly to the degree of very large numbers, numbers like a Googol (not the search engine, but 10^{100}), very big numbers that say we're really not likely to find it anywhere else other than where we think we're going to see it, but "not likely" and "not" are terribly important distinctions with consequential physical manifestations under quantum conditions.

Everything has a wave function. Schrödinger's equation doesn't just apply to basic particles like photons, it also applies to atoms, molecules, and objects. We can compute the wave function of ourselves or our cat or of the Earth. What we'd find is the uncertainty of the location of a cat is going to be something like 10^{-27} meters and that gives us kind of a sense of where it's going to be and not be. That is, a cat is not going to be poorly located, randomly fluctuating all over the room, at least not for quantum mechanical reasons. Something like that will be located with higher precision, by 20 or 30 orders of magnitude, than we could measure it using any classical instrument, and that's just fine for any sort of classical observations.

But it does mean is that there is a finite probability that the cat won't be here, that it'll be somewhere else, instantaneously. If we're familiar with Douglas Adams' *Hitchhiker's Guide to the Galaxy*, he wrote about how faster than light travel was achieved using the infinite improbability drive which, like much of his writing, actually comes from physics, a physics joke, sort of. The idea is that if we could, instead of trying to move things from place to place, if we could instead change the probability that they would be located in a different place, say we increase the probability that it was located some new location from 10^{-200} to 10^{-2}, then it would, in a few moments, just be there, instantly. All we have to do is change the exponent by 2 or 3 digits, not the mantissa by 100s: easy!

Obviously nobody knows how to do that as our undergarments have not leapt simultaneously one foot to the left, but it's kind of a clever idea, modifying the exponent rather than the mantissa value. And this is, at least metaphorically, how quantum computers achieve their extraordinary computing capabilities: it isn't literally true, but we might consider qubits as operating directly on the exponents of certain hard problems while classical computers operate on the scalar value. A quantum computer is an infinite improbability drive for, among other things, finding private keys used in cryptographic communications.

3.2 SPIN, POLARIZATION



Additional concepts that we'll need to work with to understand quantum computation are the concepts of spin and polarization. Polarization is a direct observable using only a polarizer, as in the standard plastic optical film, while spin is somewhat harder to observe. If you've ever had an NMR, now called magnetic resonance imaging, then spin has been responsible for us being able to get cool 3D images of our insides. These are two observables, things that we can see in the real world, that are directly influenced by quantum mechanical effects.

Spin is a subtle concept: it's the magnetic spin of a particle and can be induced to have a procession about an axis of spin like a top. But it is not like a top, not really, because it can spin in any axis, indeed it can spin in all axis if in superposition. But at any time, when we measure it, it ends up being in a defined axis. Before superposition is collapsed, say by measuring it, the spin is in all axes, simultaneously. It is a complicated concept that doesn't have a clear classical analog, but most people like to think about it as a spinning top.

But despite not clearly mapping to anything we can picture, we can do some math on it by just assuming it's a spinning magnetic top and it'll work out pretty well, just remember when we talk about spin, we don't really mean spin like a top, like it's spinning at a certain speed, but spin like it can be spin up or spin down and in all of the three mutually orthogonal axes until we measure it. If we measure it and find it is it in one axis, then it is not in any of the others. It is all or one.

It is by taking advantage of the spin of an atom, or the spin of the electron field surrounding a nucleus, that we can do cool things like NMR, wherein the collective spin orientation is a function of the local magnetic field, and the magnetic field strength holds a field of tops to an orientation. Furthermore, it's the equivalent of gravity orienting a toy top: and we can hit it with a pulse of energy, like if we have ever flicked the top, it starts to processes about its axis as a gyroscope, it doesn't just fall over.

Like a top precessesing about an orthogonal axis while spinning around the axis of rotation there's an analog to this in quantum computing and in NMR in that if we hit spinning atoms at just exactly its precession frequency they will precess with larger and larger amplitude in resonance until eventually they'll try to go horizontal, which a top can't because it'll run into the edge of the top, but there's no such limitation on a spin of a particle, so it can continue to absorb energy from the field at resonance which goes into driving the amplitude of precession.

And so by putting pulses of energy into a field of atoms that matches exactly their harmonic frequency the atoms will draw energy out of the field, while when the pulse frequency isn't at resonance it can't drive the system as efficiently and so there's less loss of RF energy. In NMR the RF field that we're using to tap it, the frequency of the radio pulse is the tap frequency, we measure that draw-down in energy being converted into the precession motion that peaks at exactly the resonant frequency and from that we can estimate the number of tops absorbing that energy of being driven to precession and at what frequency; and that's what an NMR machine does and that's also how NMR-based quantum computers work.

In each case, the harmonic frequency is effected by very subtle changes in the amount of energy that that is absorbed based on the local atomic conditions influencing the atoms under test as they are driven into procession in order to determine something useful. It may be about where that molecule is located, for example by imposing known gradient fields one can determine where an atom that should respond to a specific excitation frequency given a specific magnetic field strength is located in 3D space. Or one can determine what a specific atom is chemically bonded to if we're trying to do chemical assay NMR, where the chemical structure influences the precession harmonic frequency.

Another concept that's useful is polarization, something that most people are quite familiar with. The LCD screen you're probably reading this on is based on polarization and so modern displays rely on quantum mechanics. The gray-scale that you see is a quantum effect.

If we take two polarizers and orient them at 90 degrees to one another no light passes through, it creates a black pixel on an LCD screen and if oriented in the same direction light gets through and we have a white pixel. Light from the back-light or projector bulb gets polarized in one axis then it hits a second polarizer (often called an analyzer) that's rotated 90 degrees and is blocked, so no light comes through or it's not rotated and it does go through. Pretty straightforward.

But if we rotate the analyzer 45 degrees as shown on the slide, or if we put a 45-degree rotated polarizer between the first and second polarizers (not shown on the slide), some light does go through but not all of it. That is, light is polarized by the first polarizer, then hits the analyzer 45° and so, with some probability which is a function of the angle between the two, is either blocked passes through at rate that is some function of the angle between the two and so we will see a dim (or gray scale) pixel.

What's tricky about that, what's subtle about that is that photons are particles and should either go or not go, they don't get individually split in half, so some portion goes through and some not, but each photon either goes through or not. Why?

What's happening? How do they choose?

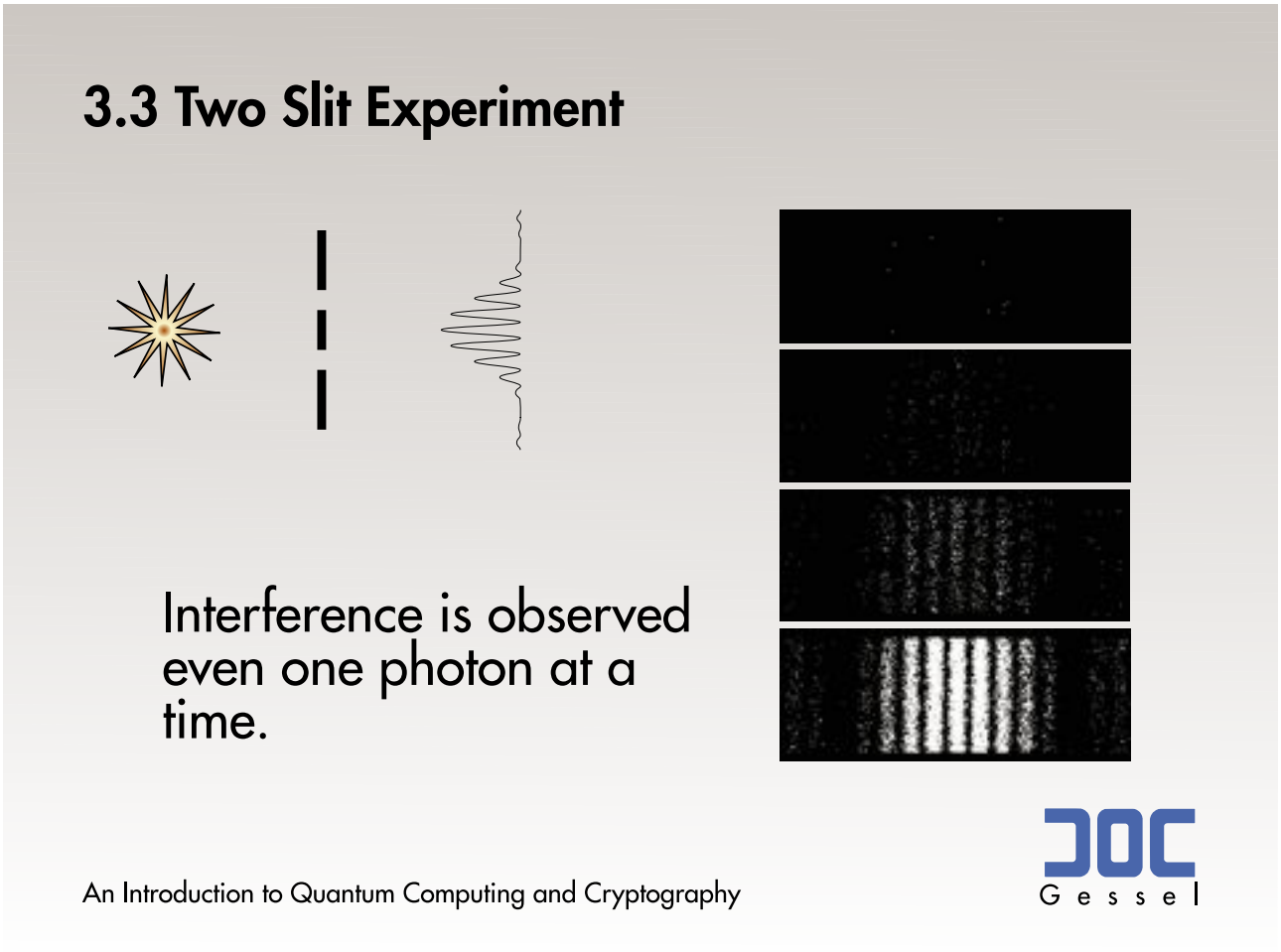
If we think about light purely as a wave it makes sense, and that is how we work out the equations. We know that this works even if we shoot a single photon at a time through polarizers, really a particle by particle experiment, we'll get 50% intensity through a 45-degree rotated polarizer. This means some photons go through and others don't, 50-50, completely random. Even one by one, some go through and some don't.

That's very subtle.

Why does that happen? Are they slipping through? Are there defects that let some through? Is there a threshold effect?

There are not. They're all being affected exactly the same by the same polarizers. It's not local imperfections or perturbations in space-time nor is it random chance that allows some to go through. 50% do something particles shouldn't but waves should. This is an accessible example of wave particle duality. This concept of the wave nature of particles, indeed of all matter and energy, is intrinsic to quantum mechanics.

3.3 TWO SLIT EXPERIMENT



If we create a narrow slit and shine a light through it at a target, so we can measure the brightness as a function of distance away from the slit in either direction, we'll get a function that looks a little like the wave function earlier, a bell curve, as if we drew a line across the peaks of the wavy function shown above. That's what we'd expect: brightest in the center and fading out towards the edges, as most of the photons go straight through as if they were bullets and hit right on center, but some go wide, fewer and fewer the further we get from center behind the slit.

But if we instead create two slits the light that reaches the target forms a complicated function because the light from each slit interferes with light from the other. That makes perfect sense from a wave equation. We can see something similar if we fill a dish with water and put a little wall in it with two isolated breaks then make a wave on one side that splashes against that wall. Each gap will emit half-circular waves that will interfere with each other and where the crests meet the joined wave will be twice as tall, or twice as bright, and where a crest meets a trough it'll cancel out and we'll get nothing.

Wave interference is something that seems familiar and reasonable, but what's weird is if we do this a particle at a time, it still works. The black images are actually from a single photon counting system which detects photons one by one. Then the top image is formed by only 10 photons pass through one or the other slit and the image clearly isn't a normal distribution.

In the second image, 100 photons have been detected and we're starting to see a pattern in the image. In the third image, 1000 photons have been detected and we can clearly see a clear diffraction pattern being built one photon at a time.

Now, what's strange and what's even more odd is if we block one slit, we go back to the regular, single peak, pattern that we get from one slit right in the center fading off to the edges, no interference pattern; if we unblock the slit we go back to the wave interference pattern.

How is it that photons going through one slit know what's going on with the other slit? How can one slit's opening or not affect the landing spot of a photon that goes through a completely different slit?

It gets even more weird in that if we could watch which slit each photon went through, if we could see the photon as it went through, then we would know which slit it went through, and we should get two bell curves not an interference pattern, one peak for each slit, with a detection peak right behind each slit and fading smoothly off with distance.

We should be able to count the photons as they go through; we should be able to fool this function. So why can't we just put a little side detector at each slit and watch as the photon comes through, and then we'd know which slit it went through and then figure out how there's a diffraction pattern. Why couldn't we do that?

Well, what happens is if we watch it, any way we watch it, requires a certain amount of energy be put into the system. That is, we can't observe the system without changing the state of the photon as it goes through the counter; to reference a common misconception: to "observe" (a bad translation, what matters is being able to measure the state of the photon) intrinsically requires interacting via enough energy to extract information from the photon, which is at least just exactly enough to change the photon's behavior from wave-like to particle-like. When we observe it, or measure it, we change it such that the pattern formed changes from a wave function to a particle function.

Any way of observing it, any method of observing it, disrupts it enough that it will change from an interference pattern to a particle-like two-mounds pattern. What's even more interesting, at least philosophically, is that, as I said before, the wave function applies to everything: objects like a cat, and in fact, people have proven the same diffraction pattern occurs when we fire something as honking huge as individual atoms through slits, not that one should test this with cats 🙀. This is very odd indeed because we can definitely see atoms, but the entanglement is so delicate that the energy to see them collapses the wave function. With photons, we always talk about wavelength and we're pretty comfortable with the idea that a photon is described by a wave function. It's got a wavelength. That'll make sense. But atoms? What's the frequency, Kenneth?

But if we're somewhat comfortable with the combined concept of photon as wave and particle; most classical behavior of photons is described by wave equations after all, but atoms are clearly particles that interact with each other, they're "things," not waves. Things we touch and feel and make cats out of. We describe atoms as particles.

Well, it turns out that if we don't disrupt the atomic system and we fire atoms through a pair of slits, if we keep the energy of disruption low enough that we don't cross the threshold, which would be the observation or measurement threshold, that is the amount of energy that it would take to observe one atom moving through that system, then we'll form an interference pattern, just like we do with two slits and light.

If we observe at all, inject just enough energy to observe, interference goes away. We get two mounds of atoms past the slits. The strange conceptual step we have to take is that, in fact, the photons or atoms or cats in some way go through *both* slits.

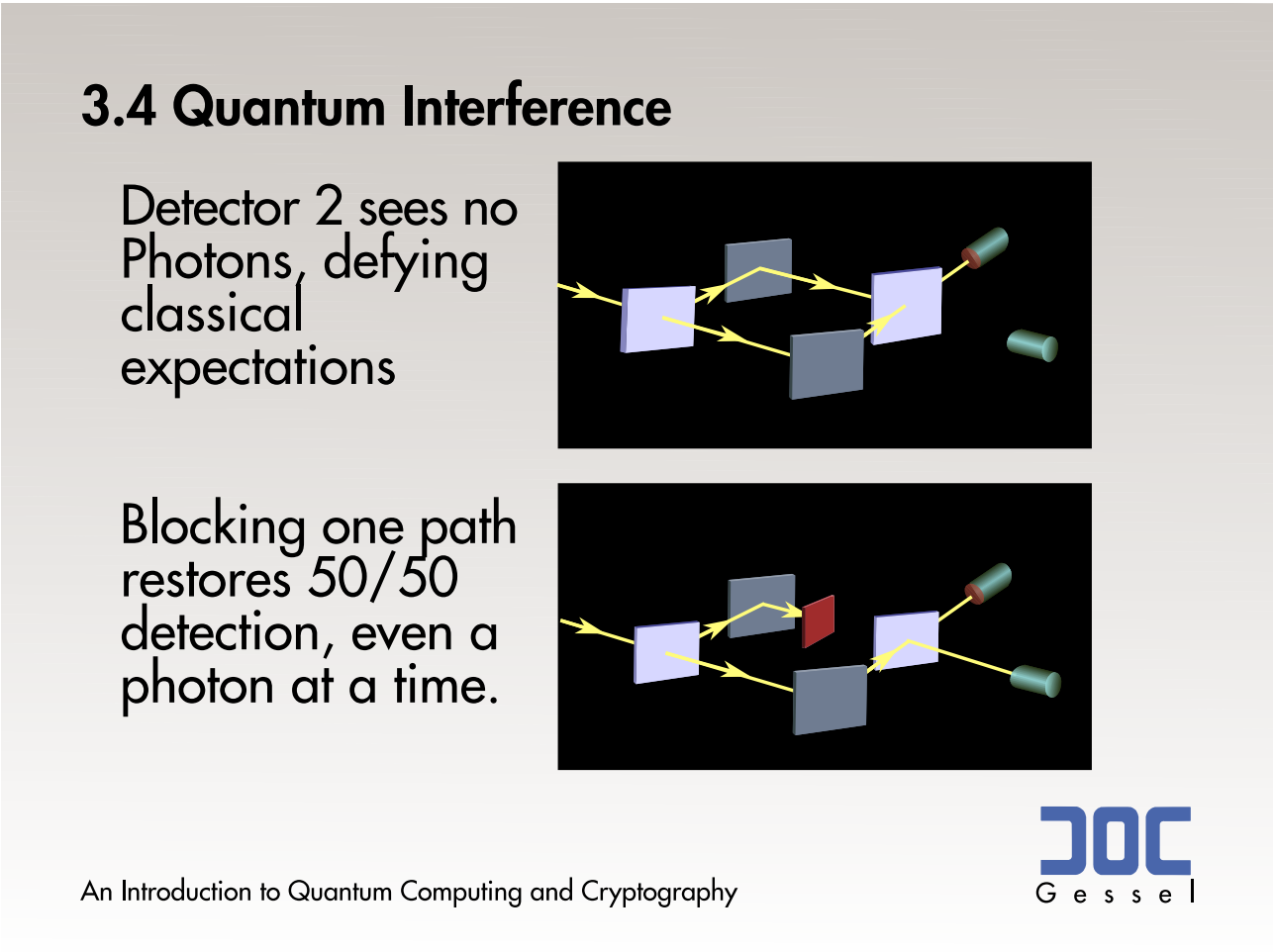
Each one, each cat, atom, or photon, goes through both slits. And the reason each one goes through both slits is non-locality, that an atom isn't in any one place at one time it's actually spread out across space with a distribution described by Schrödinger's equation. That spread of non-locality, if it's larger than the spacing on the slits, will form an interference pattern when the probability waves pass through both slits and interfere with each other on the other side of the slits and then combine to form an interfering probability wave of locality, which determines where the particles are detected.

So everything has a wave function, and when we build quantum computers, we'll take advantage of those wave functions, but we also have to be aware of the perturbation energy, the observation energy or measurement energy threshold. It's a problem called decoherence, and when we look into quantum computers we'll see decoherence time is one of the primary criterion of the utility of a quantum computer. The time until random energy breaks superposition must be longer than run time.

Decoherence means that random noise causes a wave-defined system to collapse to a particle-defined system and is a function of the interference of all environmental factors and system design. If there's noise in the room, if there are vibrations, if there's electromagnetic energy penetrating the apparatus, or heat and that energy passes the "observation" threshold then we'll lose our wave function and we'll be cooking with particles. We'll be in a fixed state and the system won't be interacting as probability waves anymore and it won't work as a quantum computer.

All computations in quantum computers are performed when the particles that build the qubits are defined by their wave functions and that is they're not behaving like particles, they're behaving like probability waves interfering with each other.

3.4 QUANTUM INTERFERENCE



If we set up exactly equal path lengths on a stable optical table with mirrors configured roughly like those illustrated in the slide and shine a light through the first half-silvered mirror, which will split it, then bounce the two beams off two full-silvered mirrors, then recombine them in another half-silvered mirror, we would expect each of the detectors shown to record 50% of the photons, that is half to one and half to the other.

But, if the path lengths are exact, only the top detector will see any photons, 100% of the light, as if the first half silvered mirror sorted the photons between bouncers sent up and passers sent down and once so sorted, at the second half-silvered mirror all the photons coming from the top are still bouncers and all those coming from the bottom are still passers, so the top detector gets all the light and bottom gets nothing.

The explanation is similar to the polarization model: each photon is actually traveling down both paths, no matter how long those paths are, so it's not a localization problem and something very counter-intuitive is happening.

We have a photon traveling both paths, large distances, and recombining, and still having interference effects.

If we block one of the paths, which is the lower illustration, we get 50% split of the remaining photons at the last beam splitter, which is what we'd expect. The two paths no longer interfere, so the light does not seem to carry any extra information from the first beam splitter and acts like an unsorted beam, photons not divided into bouncers and passers, so it splits 50/50. If we remove the block, the two paths interfere once again, having been sorted into bouncers and passers.

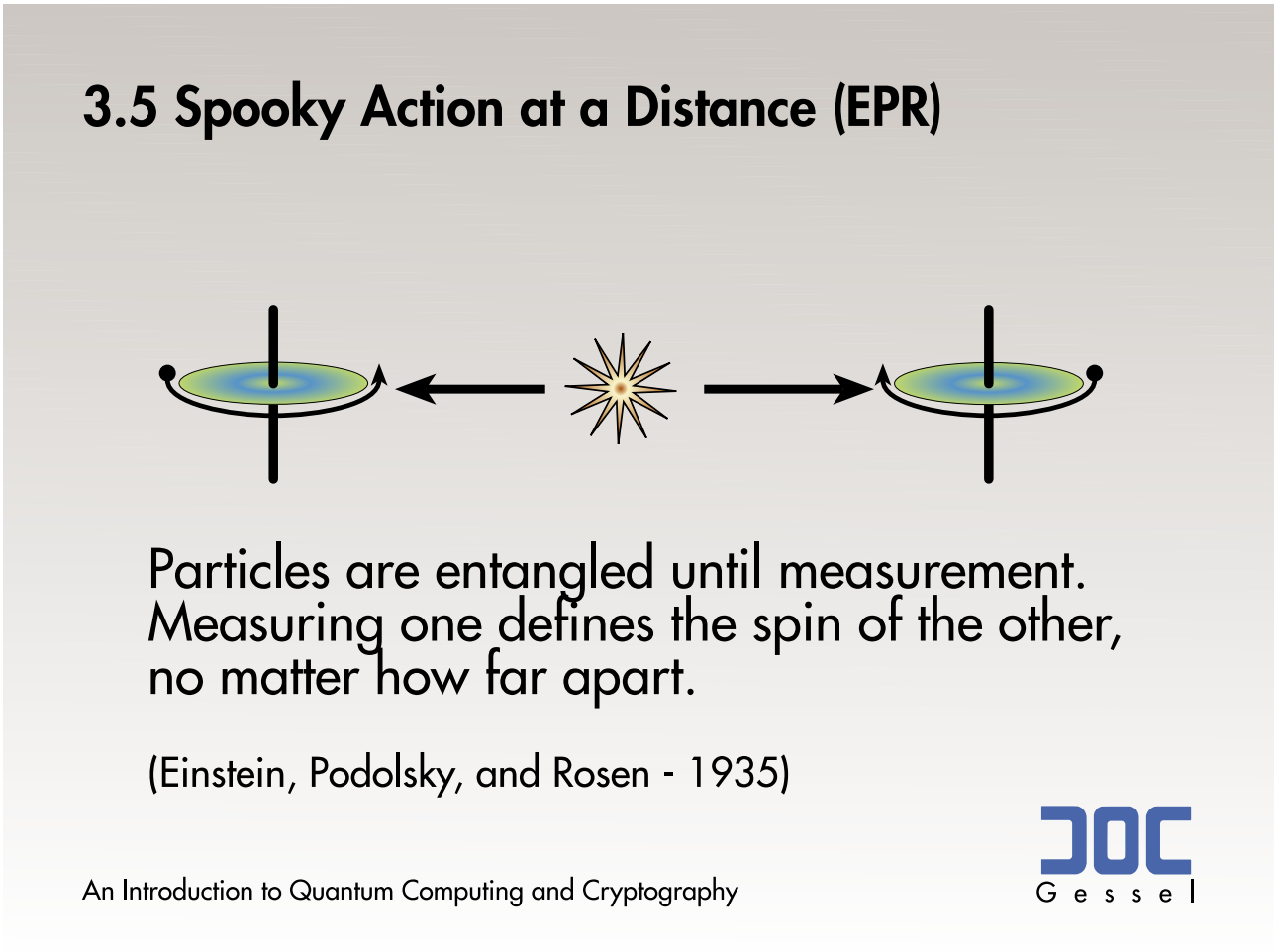
Even more interesting, if we put a sliver of glass in one of the paths that the beam has to pass through, which slows down the progress of light because light's velocity is a function of the index of refraction in the material it is moving through and glass has a different index of refraction than air so that light moves a little slower. A glass block in the path is like a time delay line for light and light slows down on one of the paths. If we adjust the thickness of glass, and therefore the line delay

between upper and lower paths, we can switch which detector records the photons, and we can pull the glass in and out to switch photons reaching one detector to the other, just by changing the time of flight along one path. It also works by slightly changing the path length, say by moving one of the full-silvered bounce mirrors, which would have the same effect.

Two beams are interacting at a level of quantum entanglement and generating macroscopic, directly observable results, albeit under very controlled circumstances.

While the wave interference effect resulting in the full beam reaching one detector or another is weird, if the bounce/pass behavior persisted whether one beam was blocked or another, it would be even weirder. There are no sorting labels applied to the photons by the first beam splitter, rather the photons are put in superposition by the first beam splitter and interact as interfering waves at the second beam splitter.

3.5 SPOOKY ACTION AT A DISTANCE (EPR)



That particles are entangled until decohered and then apparently instantly not entangled leads to a very odd inference, and this is one that really bugged Einstein, called “spooky action at a distance.” To understand better, we have to combine the concepts of wave function entanglement and spin.

In this experiment we start with a particle that has a spin of zero and will quickly decay by releasing two photons. Because of conservation of momentum, which tells us we can't have more momentum at the end of the experiment that we started with at the beginning considering losses, the decay products will also have net-zero spin.

Once the particle decays into two smaller particles going in opposite directions, two photons for example, they will each have spin one half, an allowed unit of spin. One will be up and one will be down, so the two spins cancel, as they must because we started out with zero spin and so we always have to have zero net spin.

What's interesting is that the spins have to be in exactly opposite directions to cancel, no matter what the basis angle they’re spinning around is. If the basis angle is in Z and one is spin up, the other one has to be spin down Z. If the basis angle is in X, the other one has to be in negative X; Y, negative Y. They have to be opposites in a shared basis angle because otherwise angular momentum wouldn't be conserved, but, before we measure it, which basis angle is indeterminate.

There's nothing that requires spin up, down, left, or right (or X, Y, or Z), and we can actually detect when it's an indeterminate basis state because the spin state will behave like a wave and cause interference effects just like in the beam path experiment. If we run these two particles through an interference system and we measure the interference of the two basis states, we will measure the quantum interference between them as wave-like behavior and we’ll see that they interfere as if they were still defined by wave functions. However, if we measure the spin state before we check the interference pattern, then

we will not see any interference effects just as if we'd somehow managed to measure which slit a photon passed through in the two slit experiment, which also causes the interference effects to go away.

We can prove that there's an interference effect if we don't measure either particle, but if we measure one and so know, say, that the particle going to the right is up we then know that the particle going to the left is down, so if we measure the right one but do not measure the left one, will the left one show interference effects – that is, will it behave like a wave?

It doesn't.

It can't because we know which direction it is pointing. We know because we know the direction of the one on the right and they have to be complementary. The measurement of the right particle changes the behavior of the left particle, and experimentally this seems to happen instantly, that is faster than the speed of light (not to get too ahead of ourselves, that turns out to not be quite true, but in a somewhat complicated way).

Now, this gets very strange and very counter-intuitive, and this really bugged Einstein. He thought that this implied that there was a failure in quantum mechanics, that there had to be local information that wasn't described by quantum mechanics because how can we, by measuring the spin of one decay particle at one end of the room, seemingly *ex post facto* change another particle's spin at the far other end of the room? Instantly.

This implies there's information being transmitted instantly, faster than the speed of light, across the room, that defines one particle on the basis of the other particle.

Well, it's not as strange as it sounds (nor does it require faster than light information).

There is a mathematical reason why, of course, which is that these are mathematically derived functions that are not really physical, rather a wave function that starts to expand when the particle decays, and by the time the right particle is at the right target, the left particle's wave function's furthest limit has also reached the right target and the information we gain about that wave function applies to both particles' wave functions simultaneously. We are not actually moving anything faster than the speed of light, not even information. Any effort to communicate the state difference is itself limited by the speed of light, but the wave function we collapsed came with the particle we measured at the same speed it traveled. In a sense, though the particle most likely went left like a cat is most likely where we see it, but while it's most likely location was going left its possible location distribution expanded with the right particle's movement and remained entangled until decoherence "cut" the link and the wave function's distributed existence of the particle's spin function collapsed into a known point, one explicitly by measuring it directly and the other because the spin state had been mathematically proven by the measurement of the first. The math says it couldn't be anything other than the complement to the measured particle and so all spooky quantum behaviors cease, not because we observed it directly, but because math said it had to.

This is actually useful for sending cryptographic keys. We can generate particles in the middle, and send them down both directions to people who need to communicate securely, and use spin detectors to statistically eliminate the possibility of a man in the middle attack. I won't describe the details of how the spin detectors work, rather I'll describe how spooky action works with polarization for quantum key distribution, which is the same basic principle. Polarization is a little bit less complicated because of the large number of steps it takes to measure spin is large and spin is not something we tend to interact with, while polarization can be measured with a plastic sheet, and we've all seen LCDs and so probably have some intuitive sense of how it works and what it means.

All of this is a very recent development since the early 1990s.

4.0 BASIC PRINCIPLES OF QUANTUM COMPUTING

4.0 The Basics of Quantum Computing

4.1 What is a Quantum Computer
4.2 Qubits
4.3 Entangled Registers
4.4 CNOT Gate

An Introduction to Quantum Computing and Cryptography

Gessel

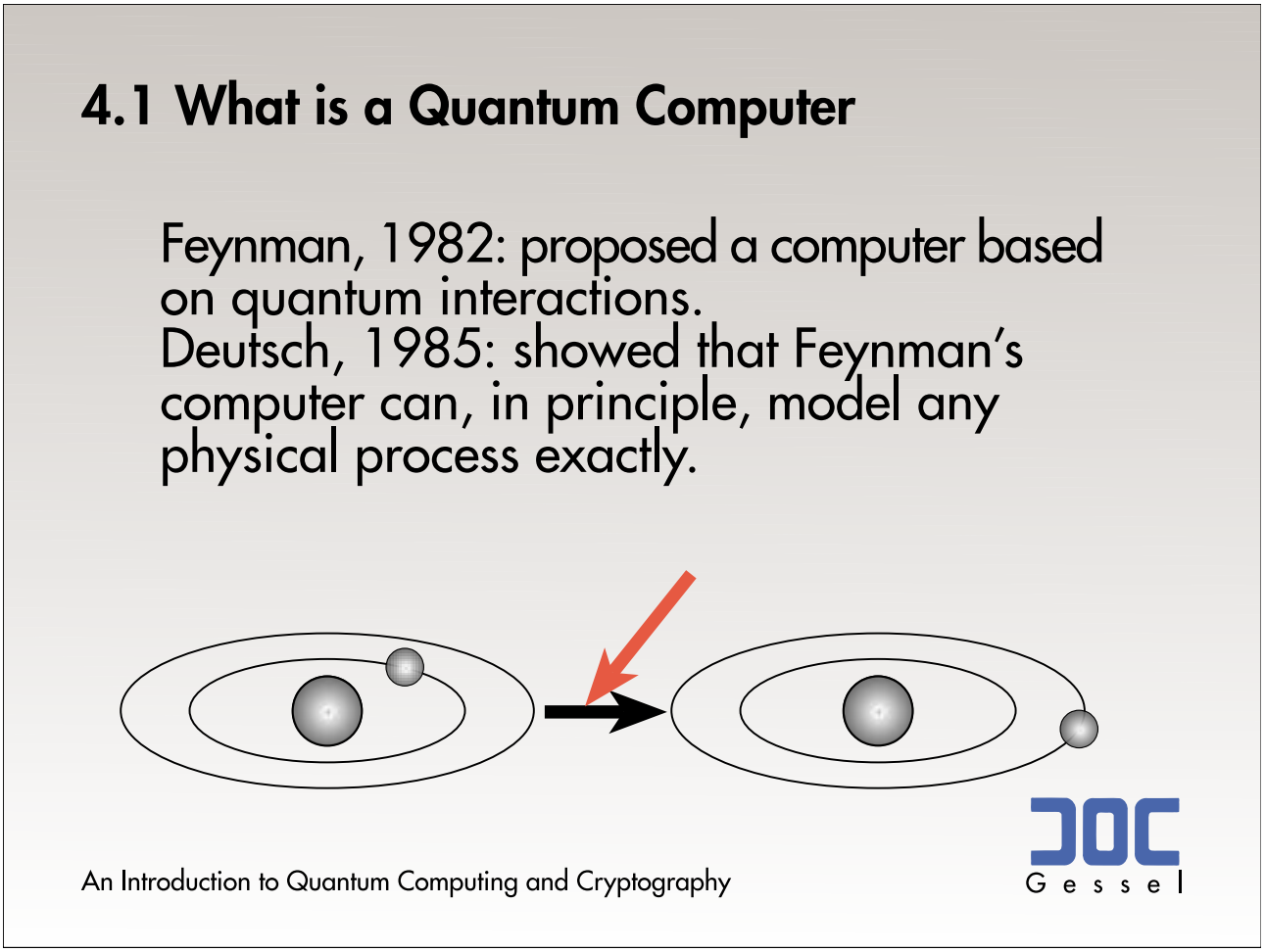
So, from the building blocks we've covered, I'm going to try to describe the basics of what a quantum computer is.

While the field of quantum computing is broad and evolving and expanding, we'll just cover some of the more universal basic elements from which quantum computers are constructed: qubits and registers and gates; just enough for basic logic.

Bits in classical computers have an approximate analog in quantum computers: "qubits" for quantum bits. Like classical computers, when quantum bits are organized, they're organized in registers. What's key about a quantum register is that it's entangled, that the bits in that register share quantum information, just like the various two photon experiments reviewed, and that has a very useful property.

With registers, we can build gates, and with gates and registers we can perform logic. And for certain logical operations, doing them with quantum registers should make nP problems solvable like P problems are on classical computers.

4.1 WHAT IS A QUANTUM COMPUTER



In 1982, Richard Feynman proposed a quantum computer as he was considering the limitations of classical computers in solving large scale physics simulations. One of the things that we might try to do with classical computers that's extremely hard is modeling the quantum mechanical behavior of multi-particle systems where the particles interact with each other, something clearly valuable for simulating the behavior of larger systems from reductionist concepts.

For example, one can encode the wave function in an algorithm and generate a graph for one entity as it evolves in time, but remember that entity affects everything in the universe not just locally, though mostly locally. Still everything in the universe is affected by the wave function of that one particle and so, if we want to describe any real quantum system accurately, we have to compute the simultaneous solution of the quantum equations for all particles which is a very, very nP problem. That is, the complexity of the problem increases exponentially with the number of particles being modeled, and it becomes intractable with classical computers a long time before an even moderately complex system is modeled.

Feynman was frustrated with this, that he couldn't simulate quantum systems on classical computers, and he proposed that one could build a quantum computer that would efficiently solve quantum problems.

In 1985, David Deutsch showed that Feynman quantum computers, through the same basic logic as Church's thesis did with Turing's machines, could solve any general problem based on analogs to classical binary bits and gates, and which could be programmed. Such a computer would be very useful for solving certain classes of nP problems by being, effectively, an intrinsically nP system.

In practice, we obviously wouldn't make a quantum word processor, it just doesn't make any sense, but we would make a quantum code breaker. We'll see that quantum computers apply most powerfully to a special class of problems, at least so far, to a very small subset of nP problems of which code breaking is one. We should, however, keep in mind this whole field

is only a few years old, so there's not too many algorithms that have been worked out for quantum computers yet. As quantum computers get big enough to be genuinely useful and more available for general use and experimentation, we can expect significant growth in the number and diversity of algorithms found to be significantly accelerated.

So, what is a quantum computer?

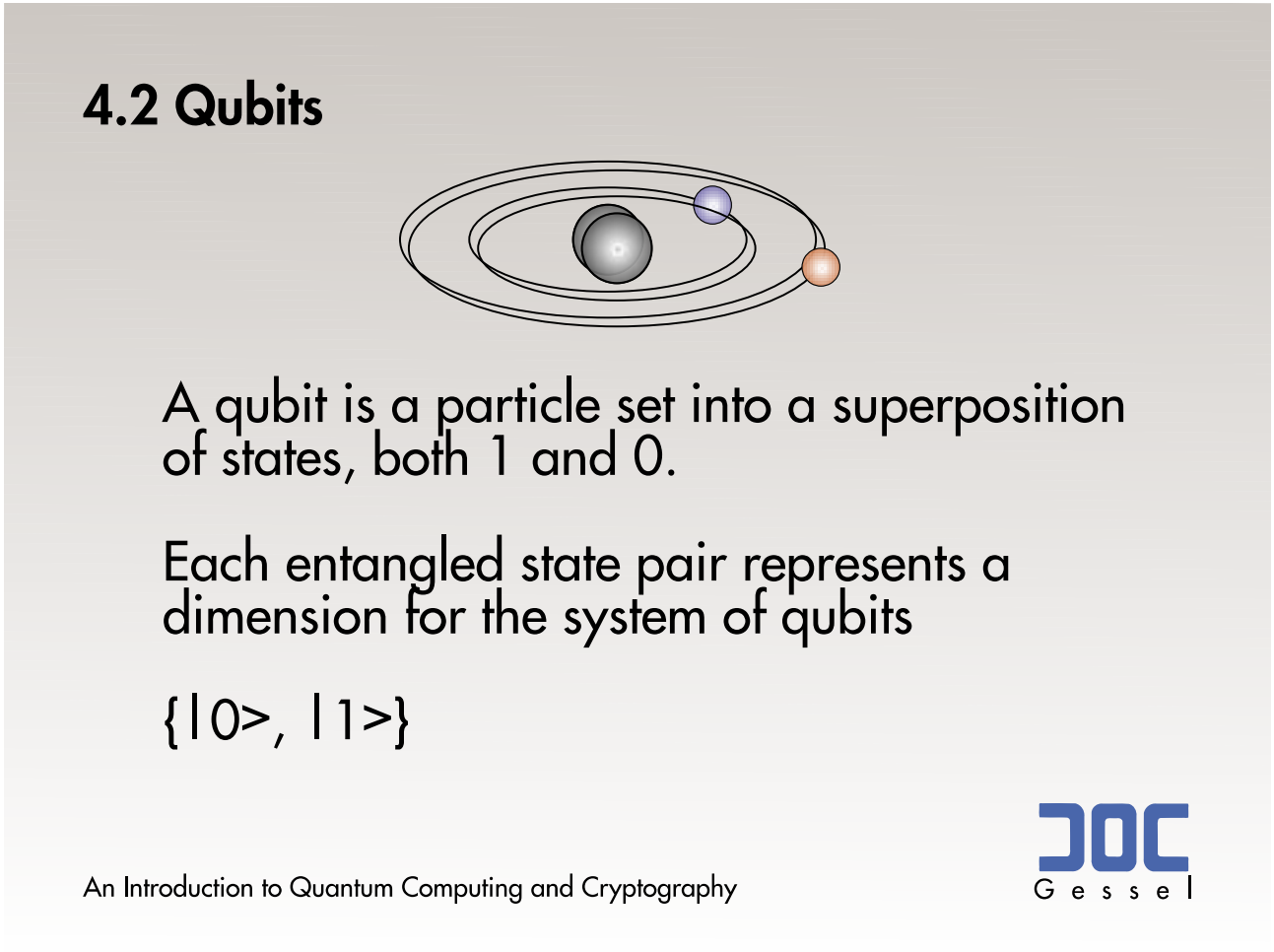
A quantum computer is, in this very simple illustration, a device that can affect a change on a particle, an atom as illustrated but any particle that has defined excitation steps. For example, an useful entangled quantum state is the excitation level of a single electron circling a hydrogen atom. A hydrogen atom's electron can be in several allowed states, that is in several different energy shells, something we might remember from high school chemistry. If the atom is in a higher state it's excited and when it drops back down, or decays, into the lower state, which it'll decay to with a characteristic time, it will release a certain quantity of energy usually as a photon or light. That's how fluorescence works, and the two states may be allocated binary values of 0 and 1, say.

A quantum computer is programmed by:

- Exciting the electrical shells into a specific energetic state as the data-input
- Setting the registers into superposition
- Running an algorithm in the form of a gate structure on the entangled register thus transforming all possible binary input values into all possible binary output values
- Running a readout algorithm that returns some value of the combined wave function that either literally contains the answer or, in some algorithms, returns a probability distribution where the answer is probably the median value.

There are a range of very different physical structures that have been built to implement this process, including ion traps, NMR, and photonic traps. The mechanisms vary but the overall conceptual steps are pretty similar.

4.2 QUBITS



If we consider a first element of a quantum computer as a single atom and where we apply a pulse to it to either bring it up into its excited state or collapse it down into its lower state, we can consider those two states, somewhat arbitrarily, as equivalent to zero and one.

By applying a half-pulse, the atom will be left in an indeterminate state, either up or down, much like a photon acts in a two slit experiment, behaving as a wave. If we measure the atoms (injection enough energy to cause decoherence), we will find that if we apply half a pulse, half of the atoms in our computer are in the excited state and half remain in the lower state, but if we don't measure it, they're in that indeterminate state collectively and entangled, at least until they decohere. Unplanned decoherence is a bad thing for quantum computers and the time to decoherence is determined largely by isolation from the environment and internal energy leaks and has to be longer than it takes to run our quantum program or we won't get a useful result.

Our goal is to achieve a superposition of two (or more) states; in this example a superposition of the excited state and the unexcited state simultaneously. This is similar to what happens when a photon hits the beam splitter in our two path experiment where our photons exists in a superposition of two states equivalent to having gone through the upper path and the lower path simultaneously. The math of that superposition is what gives us the interference effect as if the photons were waves not particles and the same math can be harnessed to similarly constructed gates to compute.

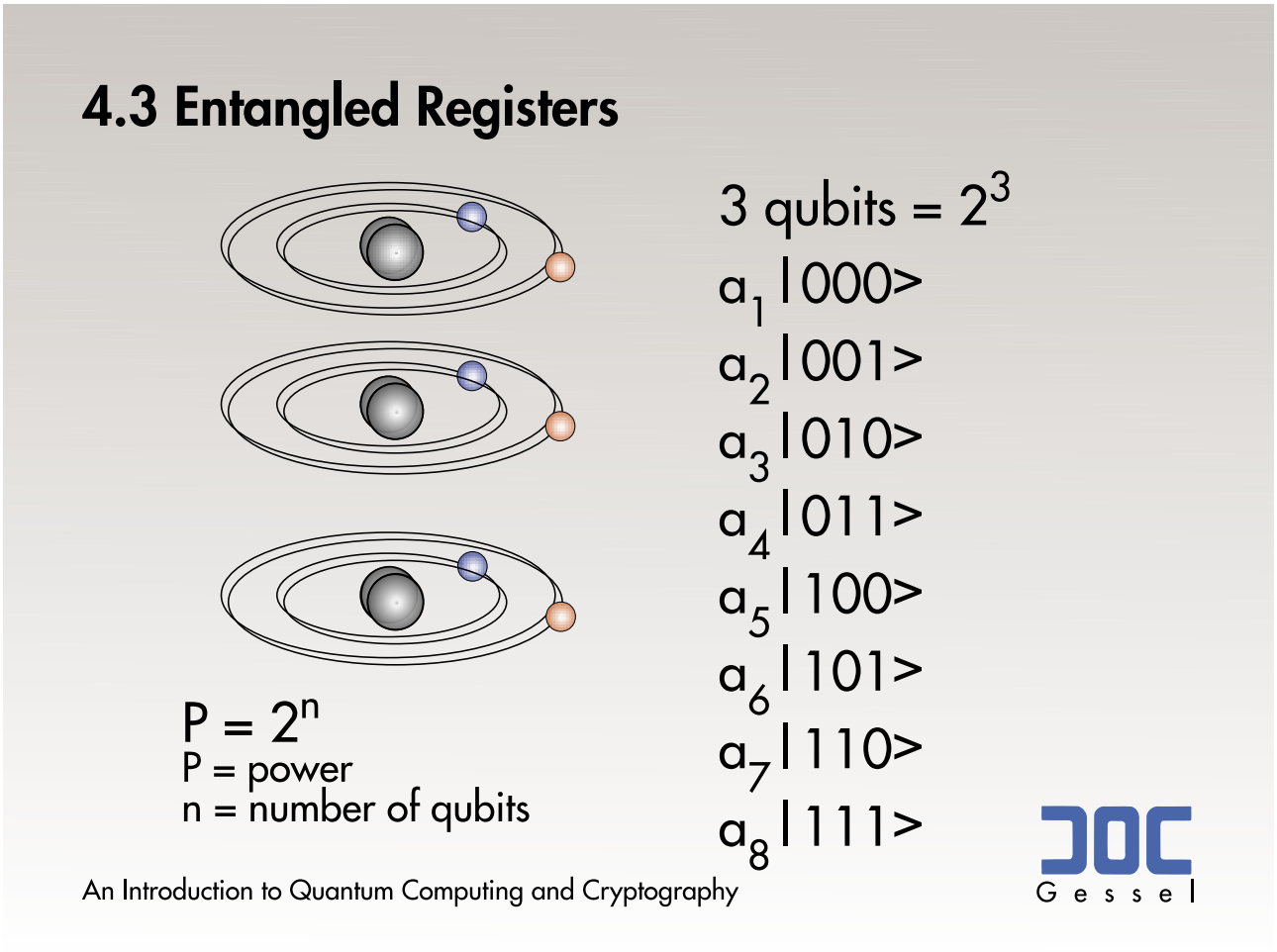
If the defining function is the wave function then there's an interference effect or an equivalent. Each qubit is a particle in a superposition of two states, one and zero and each entangled state pair represents a dimension for the system of qubits.

Now that's an interesting statement, and it comes from a fairly complicated explanation of why that is true requires a dive into Hilbert space to really explain, which is beyond this scope but an interesting follow-up study for the interested.

Staying light on the details, in a quantum computer each qubit represents a dimension of the computational system so a four qubit computer is a four-dimensional processor and eight qubit computers is an eight-dimensional processor; if not in a way we can easily picture at least mathematically. Each of the excited states is orthogonal, orthogonal in a mathematical sense, not in a physical sense. In a mathematical sense, we end up with an n-dimensional computer where n is the number of qubits in the quantum register.

In the slide there's a notation that may be unfamiliar, a pipe and right bracket surrounding the qubit variable (the glyph at the bottom of the slide). This is called Dirac notation, and if we explore the literature we'll see Dirac notation all over everybody's explanation of what quantum mechanics is. This particular glyph means the qubit is in a superposition of states of zero and one which means the particle which is the physical qubit is in a superposition of both zero and one states and not exactly either, just like the famous cat that's neither dead nor alive until someone checks 🙀.

4.3 ENTANGLED REGISTERS



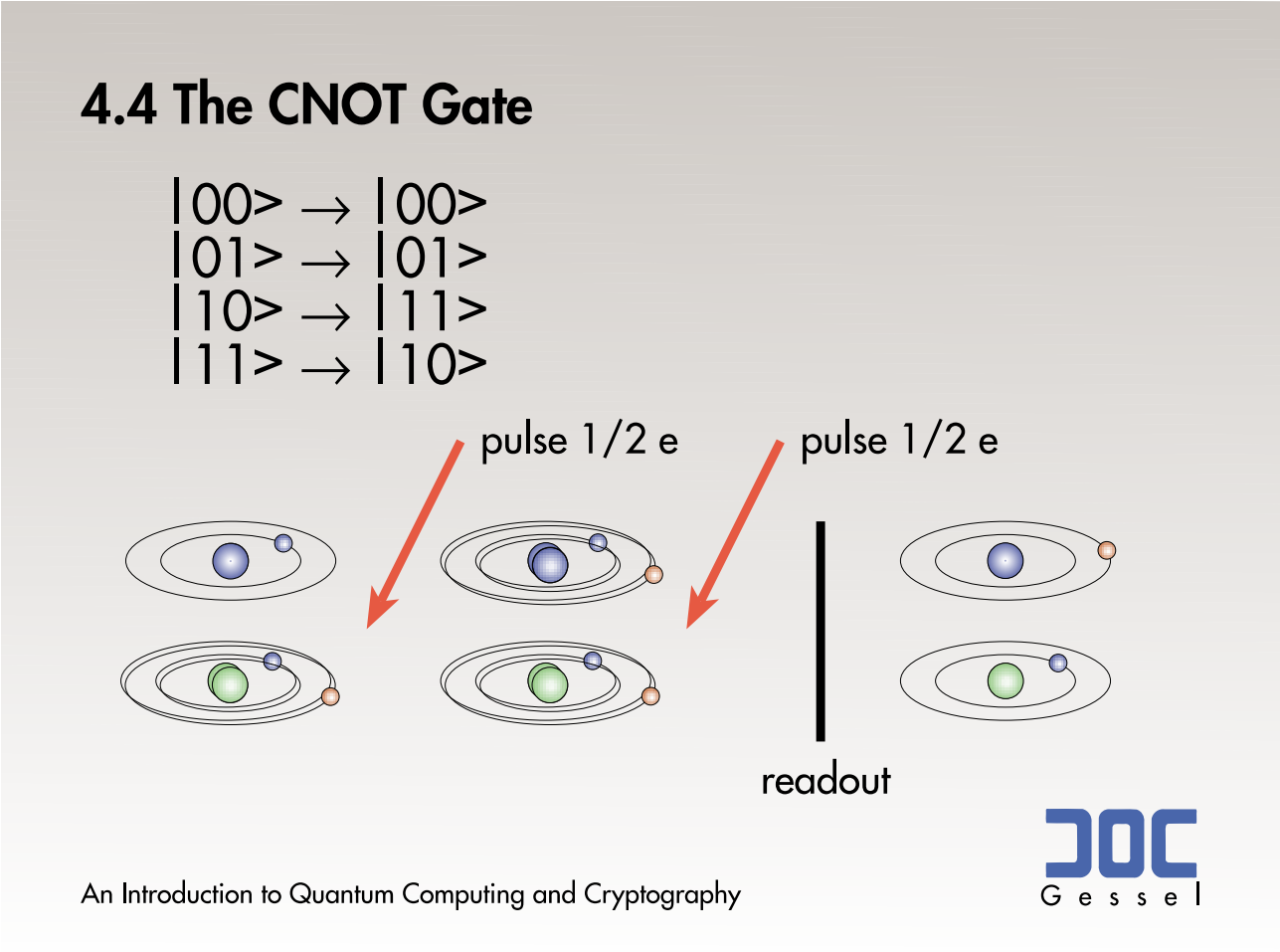
If we wanted to build an entangled register, say three qubits in entangled states, that register would encode simultaneously for *all* eight possible binary values, 2^3. This is a significant difference from a classical computer which would encode simultaneously for *one* of eight possible values.

In a quantum computer, when all the states are in superposition and functions are applied to the entangled register the operation is on the superposition of all eight states simultaneously. So what that means is that the computational power of a quantum computer increases as 2^n with n qubits for a binary quantum computer. Aside from complexity, it should be possible to build quantum computers in multiple states, ternary, quaternary, or above, for example metaplectic anyons.

Binary quantum computer increase in power as 2^n where n is the number of qubits, classical computers increase in power as the number of gates, as n. That is, power $\propto$ n for classical, power $\propto 2^n$ for quantum computers; n does not have to get very large for quantum computers to be more powerful than the largest classical computer.

In effect power of a quantum computer increases exponentially with the number of quantum elements in it that can be held in superposition long enough to execute an algorithm before decoherence erases the result. This simplified understanding provides an intuitive explanation as to why quantum computers are well suited to solving exponentially difficult nP problems like factorization in linear time where classical computers are quickly stymied.

4.4 CNOT GATE



This is an illustration of a basic quantum gate, a universal gate. The top bit in the quantum gate (blue) is the control bit and the lower bit (green) is the act bit. Suppose we have the green atom in an indeterminate state that is a superposition of 1 and 0, which we create by applying a pulse of half the energy difference between high and low, and the blue atom is adjusted into a known state, either high (1) or low (0) by applying a full pulse of the full transition energy to set it high or set it low.

If the blue atom is in the ground state, which is what's shown here illustrated by the electron being in the inner shell, while the green atom is in the superposition, which I'm illustrating by having two atoms superposed on top of each other with different states, one with a ground state electron (blue) and one with an excited state electron (brown) – the over-position is meant to indicate superposition; super, no?

A pulse of energy halfway between the two states will put both into an intermediate state, the combination of which depends on the initial energy of the green atom. That is the superposition of two states after it's been hit with a pulse of energy is not high enough to bring both into the high state or bring both into the low state, but it could go either way, and until we measure it, effectively it did go both ways – in a quantum indeterminacy sense– simultaneously.

We've put them both into a superposition, but the total energy of that system will depend on whether the blue atom was in the low state or the high state at the start.

So that's a useful property.

Now we have a system that initially we have one known and one unknown. We apply a pulse of energy, we have two unknowns, but the exact condition of those two unknowns is dependent on the control bit.

If we apply another pulse of energy, and if we knew what the control bit, the blue atom was, we could know exactly what pulse of energy to apply that will resolve both bits, the control bit and the act bit, back into their known states.

The output of the act bit will then be a function of whether the control bit was initially in the high state or the low state and so by setting the control bit, blue, we can get an output that also depends on the act bit collapse state, a logical XOR equivalent.

In the last step where we do read out, we're doing the observation step, which collapses the wave function. So by measuring the register, the wave functions collapse, and they both are resolved into a known state, the state that was read out.

Before		After	
$\|0\rangle$	$\|0\rangle$	$\|0\rangle$	$\|0\rangle$
$\|0\rangle$	$\|1\rangle$	$\|0\rangle$	$\|1\rangle$
$\|1\rangle$	$\|0\rangle$	$\|1\rangle$	$\|1\rangle$
$\|1\rangle$	$\|1\rangle$	$\|1\rangle$	$\|0\rangle$

And that's the simple gate, and it turns out that we can build, not necessarily efficiently, but we can build all necessary gates from the C-NOT gate if we apply them correctly.

There's a universal binary gate as well. We wouldn't build a computer that way as it is too inefficient, but we could if we were trying to prove it was possible.

This was first implemented in 2003 using two calcium ions in an ion trap quantum apparatus at the University of Innsbruck.

So those are the basic building blocks of quantum mechanics and the basic building blocks of quantum computation.

5.0 APPLICATIONS: CRYPTOGRAPHY, CRYPTANALYSIS

5.0 Applications: Cryptography & Cryptoanalysis

5.1 Factorization

5.2 Sieve Function

5.3 Key Distribution

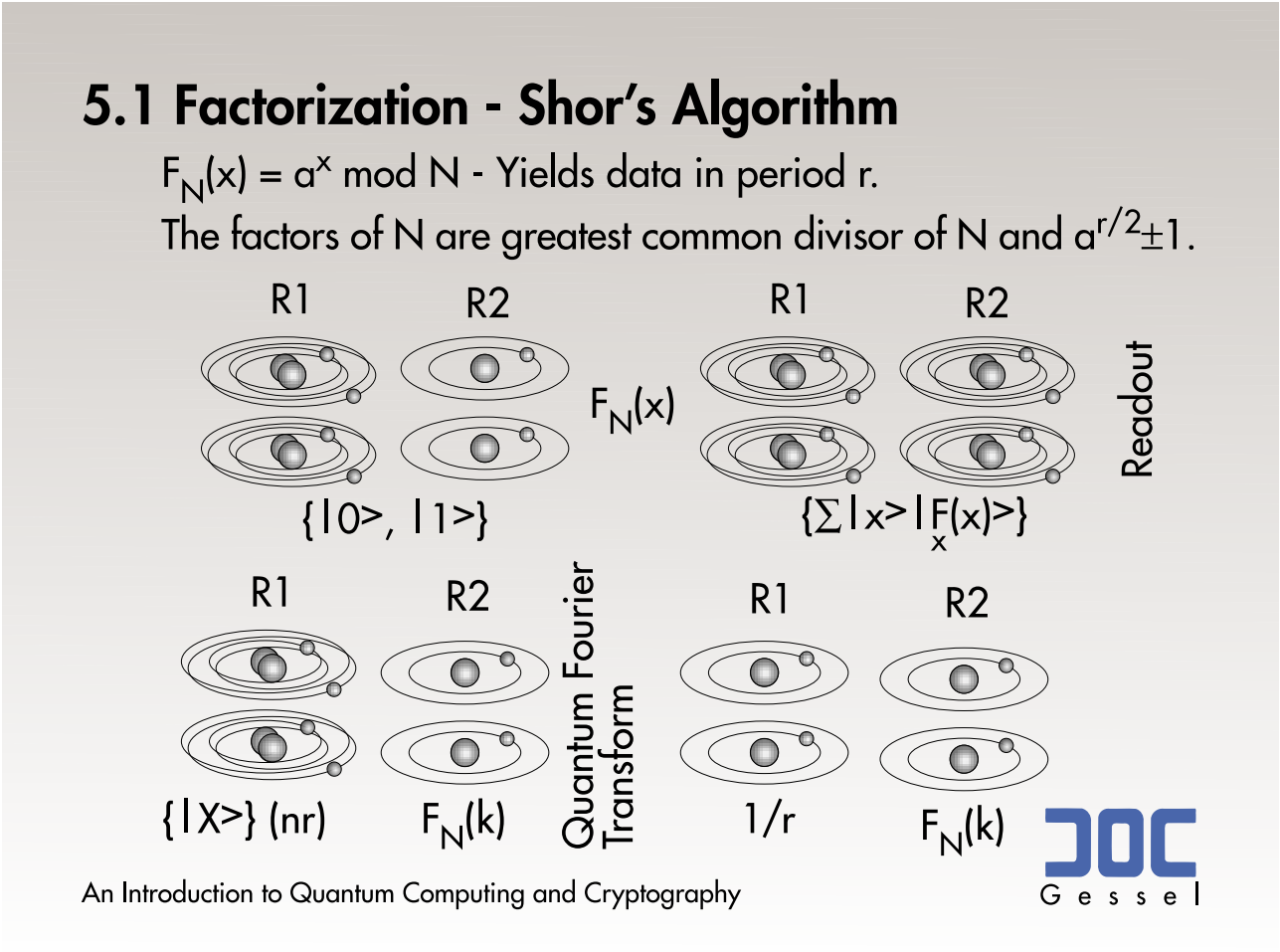
An Introduction to Quantum Computing and Cryptography

Gessel

This section will consider some applications, now that we have at least a superficial understanding of the basics. I apologize, there is a little bit of math in this. I've tried to minimize it and go through the steps as quickly as I can. But there has to be a bit of math so that we can illustrate how some parts of the process work.

The applications we consider will be relevant to decryption by factorization using Shor's algorithm which could render public key cryptography insecure, by sieve searching using Grover's algorithm which could render both symmetric key cryptography insecure and break hash functions, and how quantum mechanisms could give us a provably secure way to distribute one time pads needed for the only provably secure encryption method and the only cryptographic algorithm we can be confident is secure against quantum cryptanalysis, at least assuming we can keep the pads secure.

5.1 FACTORIZATION



Peter Shore came up with a method of using a quantum computer to find factors in polynomial time in 1994 and, as we can imagine, this is a very persuasive argument for building quantum computers because the implication of it is that if we could build a quantum computer of sufficient size and power, we can solve factorization in polynomial time, which means that encryption based on factorization is no longer secure, including RSA and other standard public key cryptography.

One of the consequences of this is that the security of all common cryptographic systems rests on the technological advance of quantum computers, and it is no longer expected to be secure until the end of the universe kind of time, depending on the rate of development of quantum computers. Given qubit progress we can predict the end date of specific cryptographic functions now considered secure, such as RSA 2048, based on empirical progress to date in quantum computers.

RSA 2048 won't be secure after 2042-01-15 at 02:01:28. Plan your bank withdrawals accordingly.

2023 update: Gidney & Ekra (of Google) published *How to factor 2048-bit RSA integers in 8 hours using 20 million noisy qubits*, 2021-04-13. This is the state of the art so far for the most efficient known (as in not hidden behind classification, should such classified devices exist) explicit algorithm for cracking RSA. The qubit requirement, 2×10^7, is certainly daunting, but with the doubling time regressed from all qubit announcements between 2016 and 2023 of 1.074 years, we can expect to have a 20,000,000 qubit computer by 2042. Variations will also crack Diffie-Hellman and even elliptic curves, creating some very serious security problems for the world not just from the failure of encryption but the exposure of all so-far encrypted data to unauthorized decryption.

With sufficient stable qubits, a fairly ambitious quantum computer still far from being realized, we should be able to crack a thousand-bit key in about a second, more precisely d^3 operations where "d" is the number of bits, so a cracking a 1,000 bit key in a second requires at least a gigaop or 1,000 quantum MIPS, keeping in mind that projecting quantum computer clock

speeds is still very speculative. Even with caveats, it is a very concerning concept with some significant ramifications and that's why I'll take a little bit of time to go through it, starting with a basic mathematical concept: modular arithmetic.

Basic modular arithmetic is written as "x (mod N)" and is interpreted as the remainder of x after dividing by N. Hours in a day is a common example: if we ask what time will it be 25 hours after 10, we might write it as 10 + 25 (mod 12). To compute this we divide 35/12 = 2+11/12, so the answer is 11 (ignoring am/pm, we would use mod 24 to disambiguate day/night.) If we were to increment the hours after 10 by unitary integers (1) we need to know the time for: 25, 26, 27, 28....; the result would loop: 11, 12, 1, 2, 3... and so the results have a period of 12, or more generally, N. Periods are wave parameters; hint, hint.

To an extremely superficial degree, most key exchange algorithms, such as RSA, DH, or EC, rely on the ease of multiplying two large primes to get a very larger semiprime number, N, a number which has only the two large primes (P_1 and P_2) as meaningful factors. Multiplying the two primes is a quick operation, a standard P problem, but finding the two primes if we don't know either is a hard problem, a classic nP problem and intractable for reasonably large numbers.

Intractable only if we don't have a decent quantum computer!

Shor's algorithm, which is based on Euclid's theorem, builds on a shortcut to finding factors that uses exponential modular arithmetic that, by some math magic, shifts the difficult step for finding factors from trial division by all numbers up to to $\sqrt{N}$ instead finding a different number, r, that has the special characteristic of being periodic and while finding this r isn't much easier than trial division with classical computers, we'll see that because it is periodic and therefore has predictable wave-like characteristics over all possible values, we can use a quantum computer to make that mathematical wave measurable and find r and thus P_1 and P_2 with far, far less effort.

We define r as the smallest possible integer that meets $a^r = 1 \pmod N$, where a is a randomly selected trial value that meets certain criteria (more detailed explanations cover the constraints of allowed a and r values). This might look very similar to Fermat's Little Theorem $a^{p-1} \equiv 1 \pmod p$, a useful way to find primes and non-trivially relevant in ways that are beyond the scope of this document. There are some good step-by-step descriptions published, but a trivial worked example is to find $7^r = 1 \pmod{15}$. If we try values for r, we find $7^2 \pmod{15} = 4$, $7^3 \pmod{15} = 13$, but $7^4 = 2401/15 = 160 + 1/15 = 1 \pmod{15}$ so 4 is the smallest integer r that solves the equation. Classically, because there's no shortcut to trying test values, finding r is still a nP problem.

This can be used to find the prime factors of a number, N, that has only two prime factors such that $N = P_1 \times P_2$. In order to do this with a classical computer, we'll use the above period calculation and try random values of a to find r for N, and from r find the primes using Euclid's Algorithm, $(a^{r/2}+1)(a^{r/2}-1) = \text{mod}(N)$, a fairly simple classical calculation once we have r and which therefore simplifies finding the factors to finding r as r will be a factor of $(P_1-1)\times(P_2+1)$.

This is significantly faster than brute force, testing all possible numbers between 2 and $\sqrt{N}$, but still an nP problem that increases in complexity as power of the number of digits. Once we find the period, r, which is the number of digits before the sequence $x^1 \bmod N$, $x^2 \bmod N$, $x^3 \bmod N$ starts repeating, we can find the factors of N.

This process is (on average) more efficient than brute force, but it definitely doesn't take factoring large semiprimes from an nP problem to a P problem, we're still talking about something that takes exponential time, so public keys remain secure. But it turns out periodicity in r is something that we can exploit in quantum mechanics in parallel, perhaps this is trivializing the underlying mechanisms, but periodicity is a wave concept, and quantum mechanics is all about the waves.

A practical quantum implementation has been found that requires 10×d qubits – that is 10,000 qubits to crack a 1000 bit key. That's awfully hard to illustrate, but we'll go through the basic steps on if we had a 4 register, 2 qubit system.

The slide illustrates an input register where we've set the values of the input register (R1) to be the superposed states of the numbers to be factored and an output register (R2) in that hasn't been read yet and is still in superposition.

Then we apply our function $F_N(x)\ a^r \pmod N$ to the register using a complicated system of gates, not illustrated; we'll just assume that we've built that set of gates about using the process described previously, and we can now apply this function from that set of gates on the registers.

After we've applied the function on the registers, we'll end up with the first and second registers in a superposition entangled state between both registers, all the qubits working together in a quantum coherent state.

Now we have 2^n computational power working on this problem where n was the number of qubits in the system, which will be a superposition of all possible states. The states on the right, in R2, depend on the states of the left register, R1, according to the function that we've applied via the gate system and therefore depend on a superposition of all values of a^r mod N.

We happen to know (or to have just learned) that a^r (mod N) = a^{r+x} (mod N), which over a values of x forms a periodic pattern in r and all we have to do now is figure out the period of that pattern, which will be the value of r, which we do, as we always do to find periods, with a Fourier transformation, or in this case a quantum Fourier transformation

The quantum Fourier transform causes the "wrong" answers, which are not in the same periodic phase, to cancel and the correct answer (probably, it's more of a stoichiometric than deterministic process) to constructively interfere and therefore by reading out the register we get r as the strongest signal.

Probably. Not always, but with a pretty high probability.

Verifying that r can be used to find the prime factors of N is computationally trivial on a classical computer and if one of several possible mathematical or quantum limitations in the algorithm resulted in an error, we just adjust our guess of a and repeat the process until we get r such that we can find the prime factors of N.

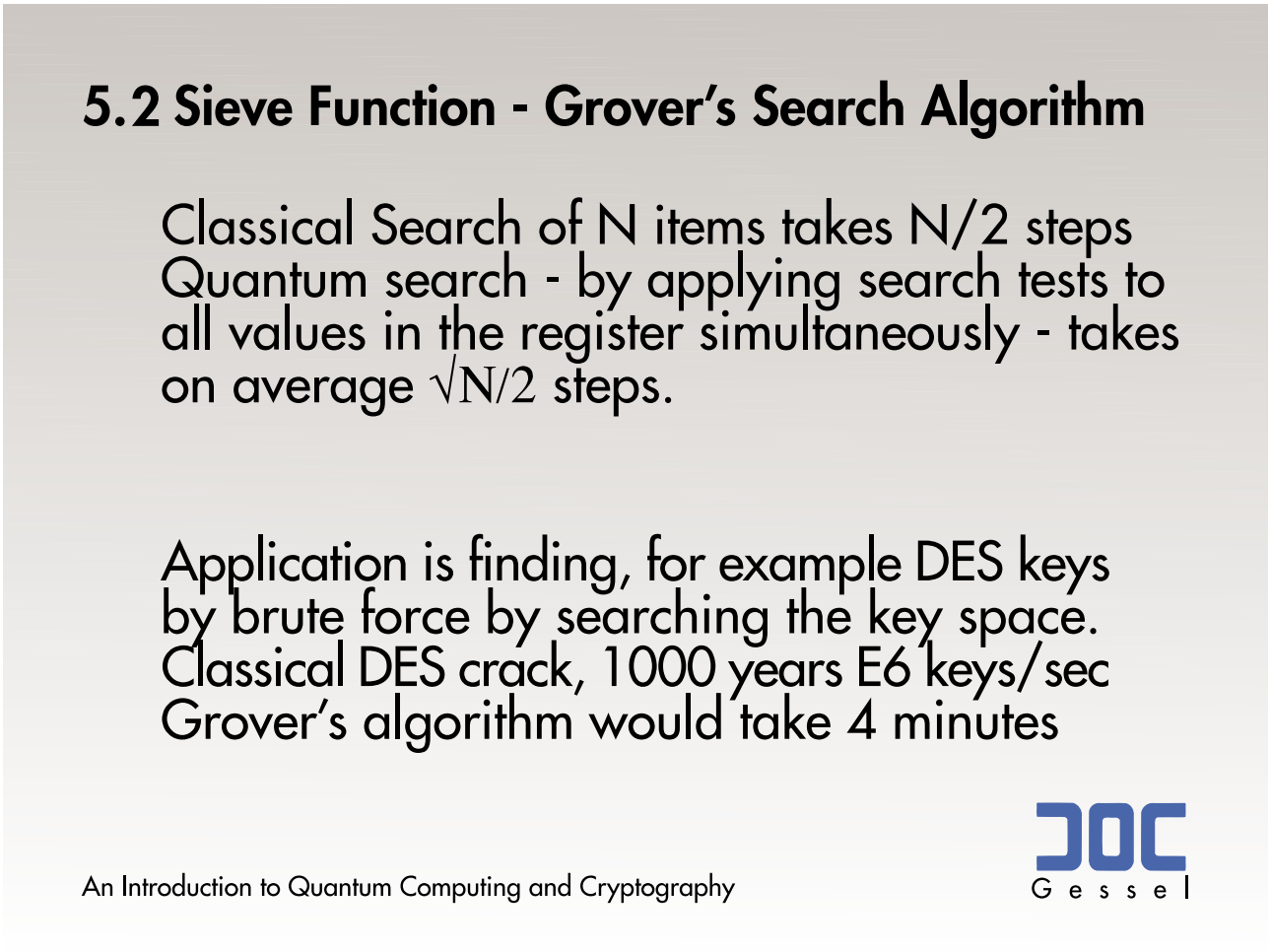
That's only four steps if we're lucky with our guess of a, or a loop or two through those four easy steps if we're not.

That's how a quantum computer can break factorization. Using a classical computer, we apply functions to bits one at a time or in groups while with a quantum computer, we apply functions to all of the bits simultaneously and if all goes well, the solution to all possible tests will be extractable from the noise in a single readout step.

Shor's algorithm has been tested in real quantum computers and it works, not for large numbers yet, but both 15 (in 2001) and 21 (2012) have been factored. You might say "wow... amazing.... I can do that in my head" and you'd be right, but the proof of concept validates the future risk (or, perhaps, present risk depending on what is running at the NSA). The big challenges to be overcome are building a computer with enough stable qubits (net after error correction which means 10x as many raw qubits for now) and with enough isolation that decoherence time is long enough for the algorithm to complete.

I realize this doesn't quite pass Feynman's test that if one really understands something, one can explain it so everybody else can, but it is the deepest we'll get into the mathematics in this paper. There's a longer, more detailed but still fairly accessible explanation from IBM on their quantum-computing site, among others that go into much more detail on how this very critical application of quantum computers works.

5.2 SIEVE FUNCTION



Another function that has significant consequences is the one that Lov Grover figured out in 1996 now called Grover's algorithm that can do a search function across all possible values simultaneously using a quantum computer. The concept is a little bit simpler than factorization, although the number of steps is actually larger than Shor's algorithm.

The basic idea is that if we're searching for something among a list, for example, we have a phone book and we know a phone number and we want to find whose name that number associates with, we have to search through the whole book for the phone number, There's no fast way to do this (without an index), on average it is going to take N/2 tries tests to find the phone number where N is the total number of entries in the phone book.

Another example might be looking through all possible combinations of some set of characters that converts a string of apparently random text into something meaningful.

In either case it'll take quite a while for a big enough data set.

If we use a quantum computer we can test all of the bits in the register, all of the entries in the register simultaneously using Grover's algorithm, it will take on average only $\sqrt{N}/2$ tries, which makes a huge difference for large search ranges.

If there were 16 items, it takes on average 16/2 or 8 steps to do it classically. When we do it quantum mechanically, it takes an average of 4/2 steps or just 2 steps to find the search term.

Now, the application of this is interesting because if we're trying to crack DES encryption, basically what we want to do is algorithmically test all possible keys and there's not a (known) factorization shortcut to find symmetric keys.

We can algorithmically test all possible keys, but that takes a rather long time. It's not necessarily an intractable problem in the sense of factorization being an intractable problem, but it's certainly a hard problem, whereas in a quantum computer, it suddenly becomes a relatively easy problem.

And in fact, symmetric key encryption falls to this algorithm.

If we take as an example a truly random 12 character password, 26^{12} or 9.5×10^{16} combinations, classically on average we have to test 4.25×10^{16} values on average to find our key. Using Grover's algorithm, we need only test 1.5×10^{8}, effectively cutting the length of our password in half, from 12 characters to just 6

2023 update: Hive Systems published an estimate of the time it takes to crack a password of a specific character length using the most powerful GPU's generally available. A 12 character password of only lowercase characters (26 characters) should take 14 hours while 6 character passwords can be broken instantly. Even 7, or 8 character passwords can be broken instantly, equivalent to 13 or 14 character passwords without Grover's algorithm. An 18 character password of all valid symbols would take 26 trillion years to crack with a state-of-the-art GPU, but using Grover's algorithm reduces this to a 9 character equivalent or just 6 hours to crack, all other things being equal.

If we try to use a classical brute force attack on a standard (now obsolete) 56-bit DES key space, it would take 1,000 years testing a million keys a second. We should note that there are standard classical computers that can do quite a bit more than a megekeyhertz now and crack DES in seconds, but testing a million keys a second is a roughly 1,000 year problem to crack 56 bit DES, while at the same basic clock speed on a quantum computer, we could crack 56 bit DES in four minutes. Note that clocks on quantum computers are not yet well-defined, so decode times are highly speculative.

Even with some uncertainty, that's a pretty significant speedup, and it really changes the relationship of security and encryption and would understandably make a lot of people very concerned.

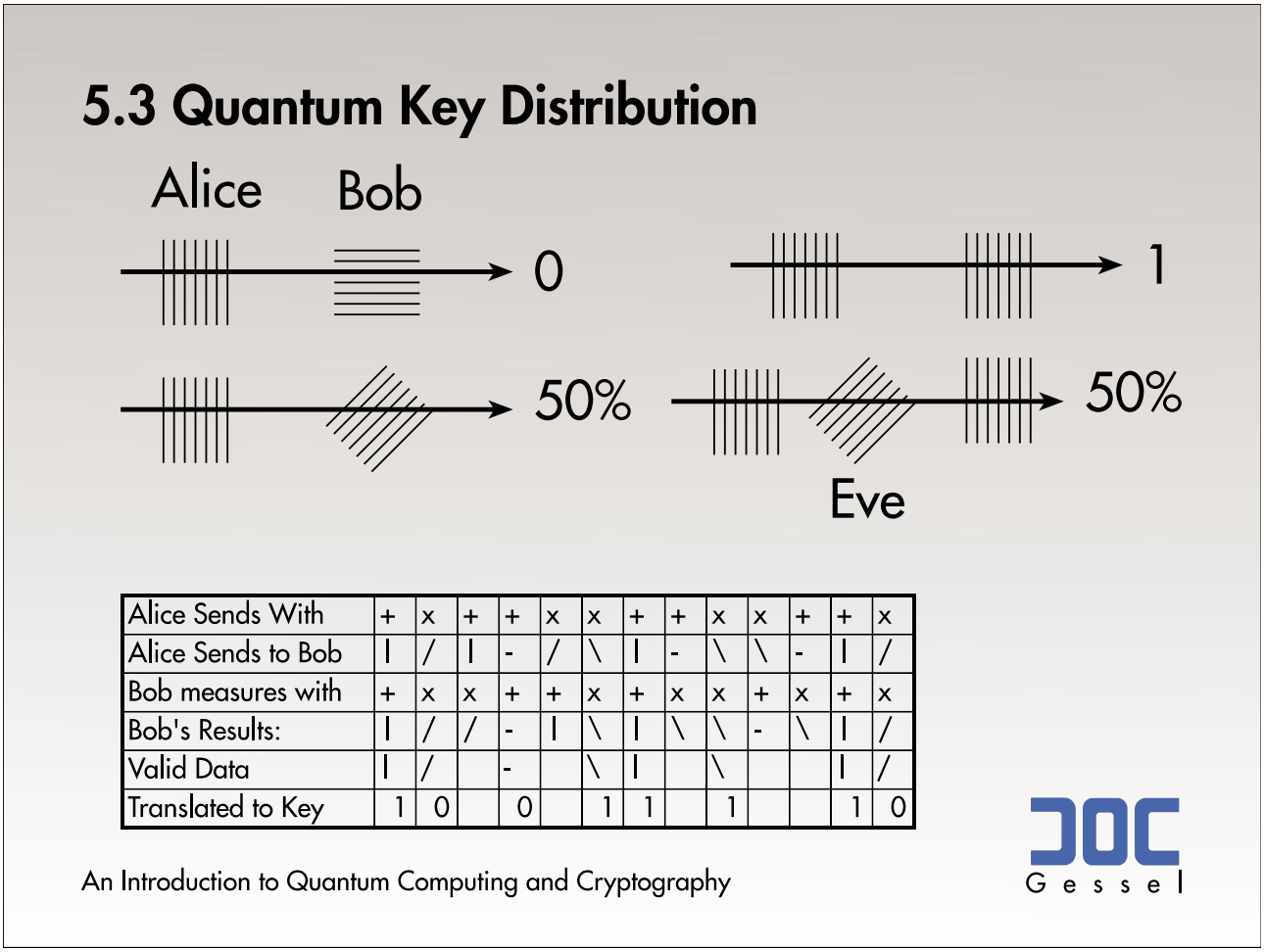
Note also that this also applies to cracking hashes and while we don't have a known quantum computer with enough qubits to hash crack SHA-256, it's naive to think this will remain true for long. SHA-256 is a 256-bit hash, but if Grover's algorithm applies it is reduced to an effective 128-bits and then SHA-256 falls and so too, eventually, does control of the blockchain. If the result of Grover's algorithm derives the public key, then Shor's algorithm can reveal the private key of any wallet. We can assume that anyone who builds such a device would quickly realize the value of quietly parsing the block chain for the larger wallets and starts quietly recovering the cost of their quantum computer and profit margin as quickly as they could before the revelation zeros out the value[1].

If a large enough government with enough resources has built a large quantum computer, they would, for obvious reasons, keep it completely secret. Alan Turing features prominently in any theoretical analysis of computers but it may help to consider the dramatization of the fate of the HMS Carlisle in *The Imitation Game* as an example of the value of keeping the ability to crack cryptography as far more valuable than the cryptographic communications themselves.

Quantum "resistant" hash functions merely mean the few quantum algorithms thus far devised don't apply, and remember we've only had quantum computers available for experiments since 2016. Assuming that anything but a one time pad is secure in a post quantum world requires a certain level of irrational exuberance.

Maybe quantum systems can help us with one time pads.

5.3 KEY DISTRIBUTION



Alice Sends With	+	x	+	+	x	x	+	+	x	x	+	+	x
Alice Sends to Bob	|	/	|	-	/	\	|	-	\	\	-	|	/
Bob measures with	+	x	x	+	+	x	+	x	x	+	x	+	x
Bob's Results:	|	/	/	-	|	\	|	\	\	-	\	|	/
Valid Data	|	/		-		\	|		\			|	/
Translated to Key	1	0		0		1	1		1			1	0

Quantum key distribution is the application that advanced most quickly in quantum cryptography and quantum computing. It's not actually a computational function and does not rely on quantum computers, it's more of a quantum cryptographic function that exploits quantum indeterminacy and by doing so provides a quantum tool to detect man-in-the-middle attacks and this, along with a simple one time pad cryptographic protocol can defeat quantum cryptanalysis.

Suppose we have the problem of trying to distribute a shared key, and let's say, the world has working quantum computers, so our public keys are useless, or soon will be, thanks to Shor's algorithm and our private key encryption is useless, or soon will be, thanks to Grover's algorithm. In such a world we're left with the only provably secure method of encryption: a one-time pad. The problem with one time pads is we have to securely distribute the pad, a simple string of random numbers known only to our heroes, Alice and Bob, and not known to our ever-present enemy: Eve. How do we do that without encryption, which is soon to be insecure?

So all of a sudden soon, we're faced with the logistics of having to use an old-fashioned one-time pad to encrypt our communication and that means we have to send big keys all over the place somehow other than over our digital networks. How are we going to get the keys from one person to another without having them intercepted? Eve is everywhere.

That's actually a very difficult problem, but fortunately quantum mechanics has an answer: we can split atoms into entangled pairs and use their quantum mechanical properties to distribute keys in a way that is provably secure; or provably not secure, an often forgotten corollary. Quantum key distribution doesn't magically protect the keys from Eve eavesdropping, but does ensure we know if she did.

This works because measuring one state collapses the wave function at both ends of the key distribution, and we can detect the collapsed wave function just like we saw with polarization of entangled particles, so although this isn't a provably secure

method of transmitting the keys because it remains possible for someone to eavesdrop on it, but it is impossible to eavesdrop on the key stream without affecting the data in a way that can be detected.

Let's say Alice prepares a pulse of light, typically one photon. We don't want to use multiple photons, but a single photon that goes through a polarizer and gets put into vertically polarization state and then sends it to Bob, over free space or a fiber. And let's say Bob measures it with horizontal polarization, then Bob's measurement will show no photon. That is a zero. But if he measures it in the vertical polarization direction, the photon will pass through the polarizer and hit his detector and he can record a one.

Alice and Bob decide on two orthogonal basis states, say upright and rotated, for polarization. If Alice tells the world, she's going to encode her bits in either 0°/90°, or 45°/135° then Bob will then measure randomly in either 0°, 90°, 45°, or 135°.

When Alice chooses to encode 0°, if Bob happens to choose 0° his detector will record 1 because it is just as Alice prepared it and 0 if he chooses 90° every time. If he chooses either 45° or 135° then he'll get either a 0 or a 1, each 50/50. That is, his answer will be right half the time by chance, but only half.

So if Alice and Bob later compare which basis states they use to prepare and to measure, which they can do publicly, but not the results of the measurements (which they can't share publicly), but sharing just basis state (upright or rotated, that is 0°/90° or 45°/135°) they were prepared and measured in they can find if there's a statistical anomaly in the error rate without revealing enough information that anyone else can find the actual bit stream.

This is illustrated by the slide table: if Alice sends with 0°/90° polarization it is indicated with a +, the table uses x to indicate 45°/135° polarization. She's preparing bits and sending them off in the basis states indicated as +, x, +, +, x, x, + and explicitly sending Bob, as shown, a 0° | polarized photon, a 45° / polarized photon, a 90° — polarized photon, etc. and these represent a string of data in the two basis states each and a 0 and 1 polarization representation in each basis state.

Bob then measures the incoming photons, but he's choosing his own random measurement basis state, + or x; there are two and so half the time on average, he'll pick the same measurement basis as Alice used for transmission and if he also chooses the same orientation he'll detect her photon. Assume Alice transmits a 0° | photon, then the probability that Bob will detect it based on how he sets up his polarizers looks like:

0° | : 100%	90° — : 0%
45° / : 50%	135° \ : 50%

To validate privacy, after a complete transmission Bob and Alice share which basis they emitted and measured in (+ or x), and where they selected the same basis by coincidence, they know that they have correct bits because the polarization will either yield a zero or a one absolutely, so they have the same bit value. Where they didn't choose the same basis state, they throw away the result because it is 50/50 that it is correct due to the 45° rotation between the emitter and receiver polarizations. What's left is a shared key.

Now, if Eve tries to intercept and she measures with her own polarizer, she could detect the photons and if she guesses the same basis as Alice, she can simply generate new photons in the same basis state she used and send it along to Bob and Bob wouldn't know.

But she has to choose a basis state to measure it in, either + or x, and half the time she's going to choose wrong but she can't know she did. She'll just randomly get a 0 or 1 output from her measurement (just like Bob does when he randomly guesses the basis state wrong). But she still has to send a photon on and Bob's going to measure it. But once she's done her measurement having randomly chosen a different basis state from Alice, she's going to create a photon in that wrong basis state rotated 45 degrees from Alice's original basis state.

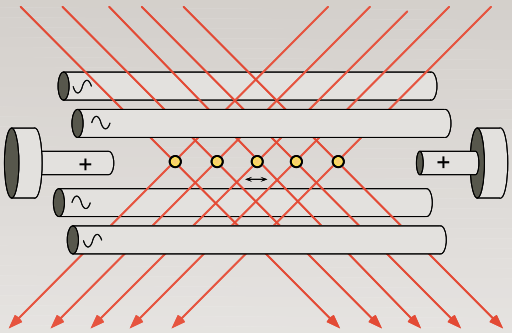
When Bob and Alice happen to agree on a bit that Eve has measured wrong by having chosen the wrong basis state, detected the orientation of the photon, and then re-transmitted it, 50% of Bob's bits will be wrong, or 25% of the total agreed bits in the key stream. As long as the raw error rate is $\ll$ 25% then this error rate will be clearly detectable, and it means that the key distribution was compromised or corrupted and we throw it away and we start over.

Or we hunt down Eve and begin again.

So that's how we can distribute now a one-time pad in a provably secure method by exploiting quantum indeterminacy.

6.0 PRACTICAL IMPLEMENTATIONS

6.0 Practical Implementations



Ion Trap Computer

NMR Computer

An Introduction to Quantum Computing and Cryptography

JOC Gessel

As of 2001, only a few practical quantum computers had been created (far more in 2023 than in 2001). One of the earliest types of quantum computers is an ion trap in which atoms are held in magnetic fields and are in excited or unexcited states, that is state zero or state one. By applying a laser or other electromagnetic pulse to inject energy one can raise the atoms from state zero to state one or by a half pulse, to an intermediate state with a half pulse and so create entangled registers and gates.

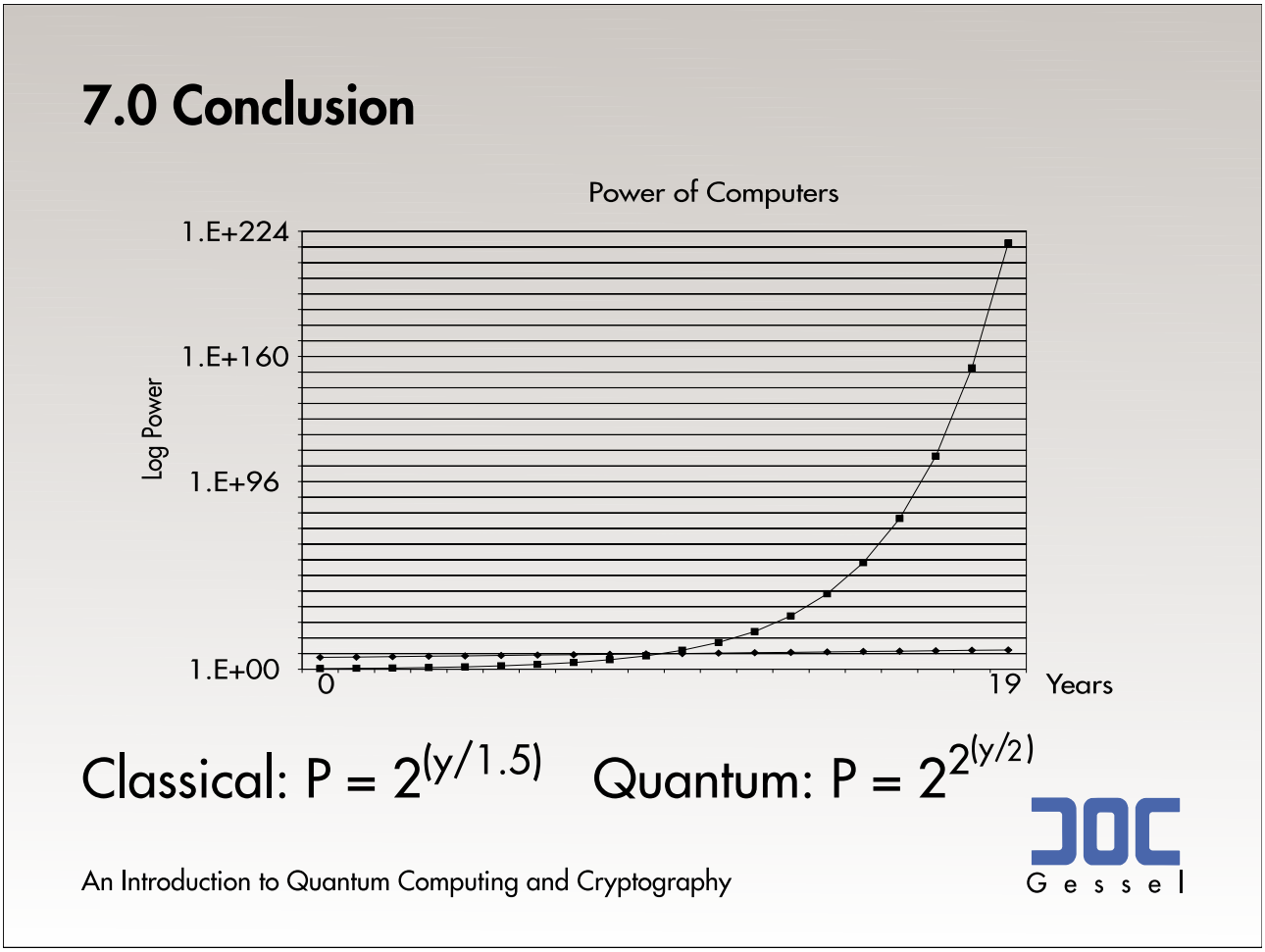
Another way that Gershwin-Feld built at MIT used NMR-based systems. NMR systems help illustrate one of the things that we talked about very briefly earlier, the rate at which decoherence happens which is the effect of the outside world injecting energetic noise into the system and which, like the energy needed for "observing," also causes decoherence.

There are two ways to slow decoherence: one is to isolate the system completely, which is what we do in an ion trap, the other is to use redundancy, which is what we do in an NMR system, where our quantum computer is actually a long hydrocarbon molecule. Each hydrogen-carbon pair is a qubit, and they can share superposition down the length of the hydrocarbon chain, which is in total a register or gate. States are set using magnetic pulses and read with others and that creates one of the types of quantum computers working today.

Neither of these look like they're going to scale to very large computers (and haven't, as of 2023), however, there's one more experimental design that was being developed in 2001 that had the possibility of being much bigger: high-Q optical cavities where each qubit is isolated in an optical cavity connected by fiber optics. This is how the largest computers in 2023 are created other than the quantum annealing computers D-Wave has been building, which are somewhat controversial.

These are still large and complex, but might be miniaturized using microfabrication techniques to build complex quantum computers of reasonable size. When this presentation was given in 2001 none existed, in 2023 they dominate.

7.0 CONCLUSION



In conclusion, what's interesting about quantum computers relative to classical computers is the exponential power of quantum computers. That is, the power of classical computers increases linearly with the complexity of processors, and the complexity of processors has been increasing exponentially according to Moore's law for decades. Since 1965, the power of classical computers has increased exponentially, doubling roughly every 18 months pretty much as predicated, though more recent growth has come from core counts per die rather than gate counts per core or increasing clock speeds.

In the graph shown, the notional power of classical computers are charted on the blue line with year zero as 1995 showing the growth in power from 1965 to 1995 which was the date of first qubit. In 1995, when there was only one qubit, classical computers had already grown about 24 million times faster than they were in 1965 when Gordon Moore proposed his law.

Since 1995, they have continued to advance pretty much according to Moore's law and extrapolating out 19 years, the graph looks pretty flat on log-linear. In 19 years, classical computers increase to something like 10^{16} times their current power, which sounds like a lot of fun if they keep proceeding at that rate.

But quantum computers have had, over the last couple of years, their own qubit count doubling rate. It's not 1.5 years or 18 months, but about every two years, roughly, we've achieved another doubling of the number of qubits that we can build from one to two to five, there's an eight qubit computer working now. [in 2020, D-wave claimed a quantum computer with 5,000 qubits, pretty close to schedule]. If that proceeds apace through the same period, in 19 years, quantum computers will be 10^{224} times faster than they are right now will be about 10^{18} times faster than the fastest classical computers.

In fact, if we project that rate out the 46 years it took classical computers to go from single gates to where were in 2001, in the same time span and starting with a single gate in 1995, quantum computers should be 10^{102605} as powerful in 2047.

That is 10 to the 102,605 power times more powerful than they are, assuming everything continues apace.

That seems utterly ridiculous; 10 with 102,605 zeros after it times faster than they are now.

That's a gigantic number.

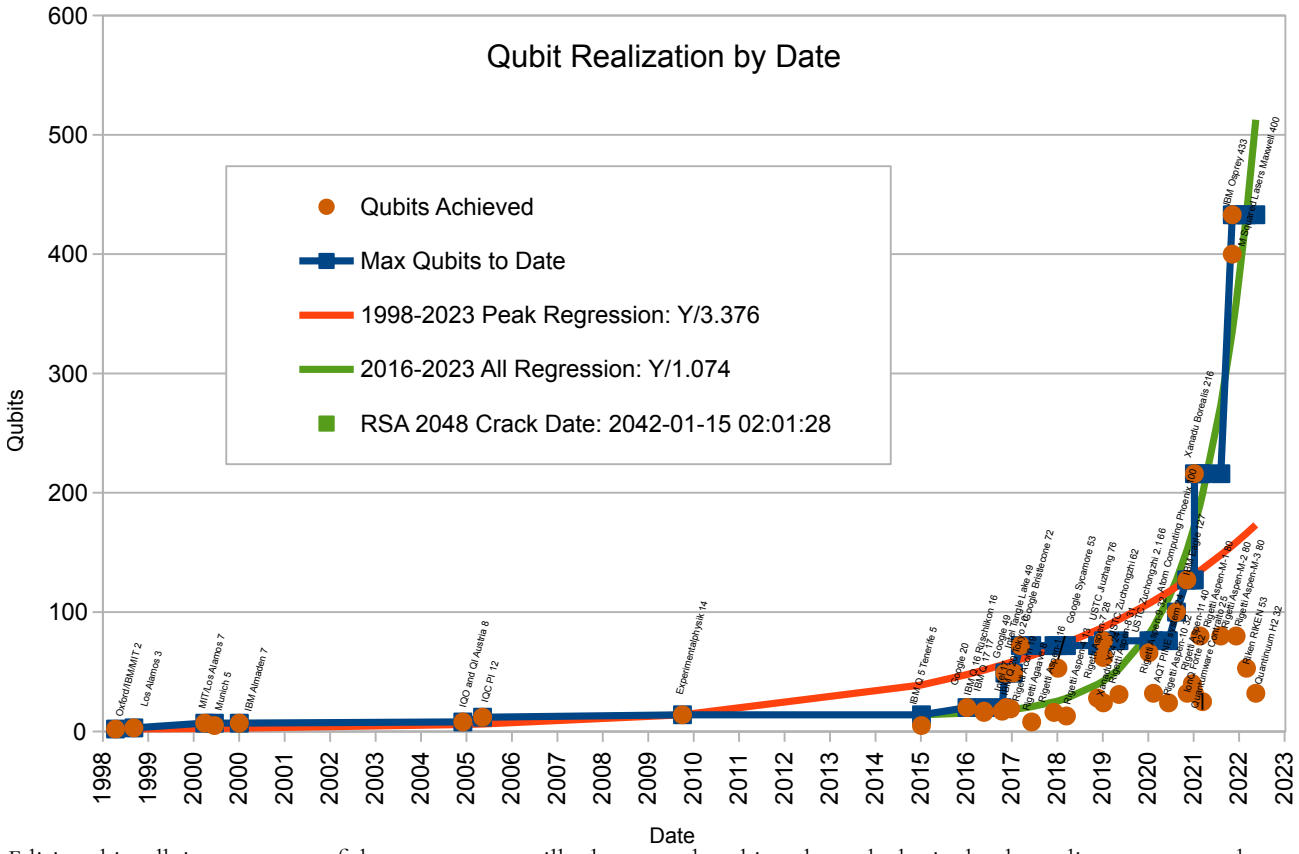
It's one of those very large numbers that becomes hard to grasp: the estimated total number of particles in the universe is only 10^{97}. So how can it be that it would be that much more powerful?

What's interesting is that while it may seem ridiculous to consider that kind of power, but remember the problem Feynman was trying to solve, the mutual interaction of multi-particle systems that was intractable on classical computers, but the universe has 10^{97} mutually interacting particles in it.

We can't build a multi-particle quantum simulation with more than about a dozen particles that computes in any sort of practical amount of time now or in any time in the reasonably expected future with a classical computer.

But a quantum computer, the universe itself, is solving that problem for 10^{97} particles at a rate of one second per second.

8.0 EPILOGUE 2023

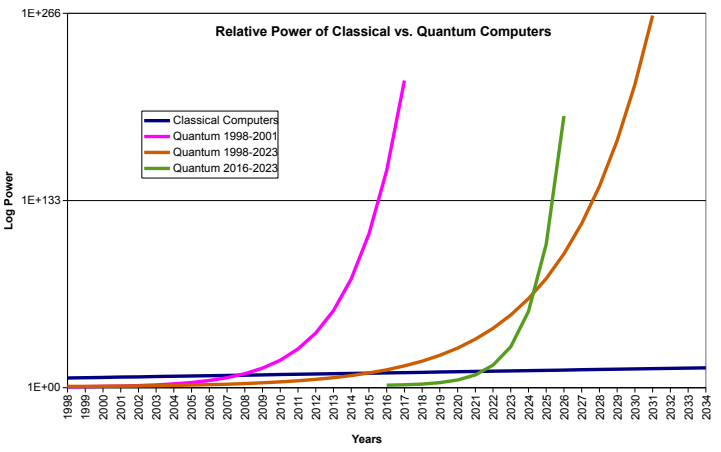


Editing this talk in 2023, most of the concepts are still relevant and nothing about the basics has been disproven, even the predictions mostly held up. Section 6, the practical implementations, is obviously very dated and historical now, not newsy.

As far as I know, nobody has taken up my formulation of quantum computing power as a time series double exponent power function of the number of qubits in a parallel structure to Moore's law. It seems compelling, despite obviously flaws. One flaw is that useful quantum computers require stable, actionable qubits, not noisy ones that might or might not be in a useful state when measured. A variety of error correction techniques have been developed in the past two decades to enable working, reliable quantum computers, but those work by combining many "raw" qubits into a single "logical" qubit, at around a 10:1 ratio.

Further, the development didn't stay exponential for long, aside from D-Wave's regular announcements of larger and larger quantum annealed systems. There was a long slow period between 2000 and 2016 when new reference machines were roughly 5 years apart. Starting in 2016 that changed and since then the number of new machines online has exploded. Not like the number of smartphones or anything, but compared to a new one every 5 years, it's a lot.

I took a regression along the full useful quantum computer history, 1998–2023 and fit that to an exponential doubling period and got 3.376 years, quite a bit slower than the heady early years' 2.0 year doubling rate. On the other hand, fitting an exponential curve to the 2016–2023 period yields a doubling period of only 1.074 years, and only 0.820 years if we fit to only the most powerful quantum computers released, though this seems a little statistically suspect.



Just for fun, based on the 2016–2023 all data regression and a 2021 paper, we predict RSA 2048 to fall on 2042-01-15 at 2am. As a data check we predict "Quantum Supremacy" right at Google's 2022 announcement.