Internet Security and Quantum Computing

Hilarie Orman
hilarie@purplestreak.com

December 13, 2021

Abstract

The cryptographic algorithms that we rely on for Internet trust and security are based on the computational difficulty of solving particular mathematical problems. Sufficiently powerful quantum computers could solve those problems in a day or less, rendering their protection largely useless. When this hypothetical future is on the horizon, Internet software suppliers should undertake the massive project of changing the fundamental cryptographic algorithms to completely different kinds of computations. This paper calls attention to how today's algorithms could be vulnerable, to the factors that impede the realization of quantum computing, and to how technologists might measure the distance to the quantum horizon.

1 Introduction: Waiting for Qubit

Someday there will be quantum devices that will solve hard problems that are unapproachable today. Even though this technology is in the distant future, now is the time for Internet security technologists to learn how to evaluate and respond to advances in quantum computing. No rational plan for long-term security can ignore it, but its impact on short to medium term planning should be modulated by the reality of the technology.

We already know a lot about how those future computers will work, the algorithms they will run, and how we will program them [5]. But none of this knowledge is enough for us to predict when these quantum computers will be built. Yearning for the future of computing is not enough to make that future real.

People who were born before the middle of the twentieth century have seen the amazing set of discoveries that began with the invention of the transistor and ran pell mell through to today's billions upon billions of computing devices and supercomputers solving many of the problems we once thought were forever elusive. We might be tempted to think that this is the established path of technology development. Surely we can start with some simple quantum device and exponentially build more capability, resulting in a quantum computer in every smartphone within the span of another lifetime?

The reality is that quantum devices are much harder to build and control than are semiconductor devices, and getting onto that exponential curve involves climbing over some very steep barriers. Quantum computers, as desirable as they are, are not "just around the corner."

The anticipation of eventual quantum computing is tinged with apprehension about its impact on Internet security. Although there is no overarching security plan for the Internet, there are essential cryptographic security mechanisms that underlie almost every communication used by websites and applications. Most communications are encrypted in order to provide a minimum of secrecy (privacy), and the way the encryption keys are provided depends on algorithms that could be rendered useless by sufficiently powerful quantum computers. The way that entities (websites, for example) are authenticated similarly depends on algorithms that are vulnerable in the possible quantum computing future.

Because of this apprehension, NIST and NSA have been investigating cryptographic algorithms that would remain secure even if large-scale quantum computing were a reality [2] [6] [32]. There is general agreement that switching to new algorithms will be inconvenient at best and disruptive at worst (regarding even a small modification to an existing protocol, see [24] [32]). The question of when to incur this cost depends on predictions of when quantum computers will be part of a mature, scalable industry. As with any attempt to predict the future, there are wildly varying opinions.

A few years ago the National Academy of Sciences assembled a team to summarize the impacts of and prospects for quantum computing [38]. Their report constitutes a good framework for thinking about quantum technology at that point in time. It also points out the breadth of concepts that underlie the field. Any computer professional who wants a timely overview will find it challenging to follow the scientific literature or even to evaluate the high-level summaries written by the technology press. What we address in this paper is the relationship between today's quantum computing research and long-term security needs of the Internet.

2 What the Internet needs from Cryptography

To understand how to approach cryptography in a quantum future, one must understand that the Internet was built without a security plan. The early developers saw security as a barrier to rapid innovation, and they believed that it would be counterproductive to burden an evolving system with a constrictive security framework. When the system was mature enough for security, it was thought, the mechanisms could be added as an architectural layer over the transport layer and as features of a session layer. In short, neither security nor privacy was part of the original Internet definition.

In keeping with the long adolescence of this innovative system, there were many ways in which security was added to the mix. Standardization of security mechanisms came slowly and painfully. In some cases the requirements seemed plastic and elusive, in some cases government policies complicated the efforts. Beyond this there were patent issues and there were mathematical issues. This frothing torment began to subside about 20 years ago.

Today we have protocols for private communication with authenticated websites, virtual private networks, secure email, and many other daily activities. All of this is founded on the assumption that there are some mathematical problems that are too hard for computers to solve.

2.1 What Makes Cryptography “Secure”?

Internet security relies on the difficulty of solving mathematical problems. That difficulty is usually expressed in the amount of work as measured by the number of computer instructions necessary to unlock the secret. There is always some cost associated with a computer operation, so we can estimate some minimal cost for an instruction and multiply that by the number of instructions.

Useful cryptography must be based on a problem that is so difficult to compute that an adversary cannot possibly muster the resources to break it. On the other hand, using cryptography should be easy for those who know some small secret. Cryptographic algorithms leverage the small secret to create a huge asymmetric barrier. In general, it should take microseconds to protect a byte of data, and the cost to break the protection should be at least hundreds of years of supercomputer time.

In the 1970s, the cost barrier was about 64 bits of security. I.e., an attacker would have to carry out at least 2^{64} computer instructions to defeat the secrecy provided by an encryption algorithm. Computers and special purpose hardware in that innocent era were few and slow by today's standards. Today the security margin should be so high that an adversary would have to carry out at least 2^{110} computer instructions using state-of-the-art processing cores. That would take more time and energy than could be available to anyone on earth. But quantum computers have the strange ability to do more work in one instruction than a classical computer can. That means that the algorithms, in order to be safe against 2^{110} *quantum* instructions, have to be different. Some algorithms might need a larger key, but some algorithms will have to be discarded or replaced by completely different kinds of calculation

2.1.1 Preparing for the Quantum "Leap"

To prepare for that possibility, in recent years NIST has been engaging with the cryptographic community to vet ideas for new algorithms via its Post Quantum competition [2]. The proposed methods are hoped to be immune to attack by either classical or quantum computers (barring any "out-of-the-blue" mathematical discovery).

At this time, nothing about quantum computing is set in stone. Everything from the basic physical units through to algorithm designs is in the investigative stage.

There may be some point in trying to use cryptography to protect something for 100 years, but not for a million years. The asymmetry of effort, protect vs. break, strongly favors protect, and when discussing 100 vs. 1000 years, it doesn't make much difference. We favor using algorithms that are "impossible" to break by arbitrarily defining 2^{128} units of time or effort as the threshold point (2^{110} would be fine, but 128 is a more convenient exponent). We will see that this limit makes sense for quantum computers as well as classical computers once we define what a "unit of effort" is in each domain.

For determining cryptographic strength for an algorithm parameterized by some number of bits, we would like to determine the effort that an adversary would need to break it. The effort can be measured in time or money or resources. The time measurement has some interaction with Moore's Law. Although processors are not getting particularly faster (seemingly stuck at a few hundred picoseconds per cycle), they are becoming a lot more numerous. A machine with 1000 cores is certainly affordable by an individual, and supercomputers with a million cores exist and more are under construction today [4] [34]. We aren't sure what the limit is for number of processors. On the other hand, we can estimate an energy bound by assuming that some minimum smidgen of energy is needed to carry out a computer instruction. That minimum can be taken to be the Landauer limit for non-reversible computation, and it is about .017 eV. Theorists may argue that computation need not consume any net energy, but practitioners will note that no such computer exists today.

If an algorithm requires 2^{128} computing steps, then it is impossible to compute by any imaginable earthly device. If each step used only the Landauer limit of energy, it would need all the energy that hits the earth in 1000 years for the whole computation. Alternatively, if each step of the algorithm took only one nanosecond, and if one billion processor cores were dedicated to the task, it would take 10^{13} years to complete.

Actual computer instructions today use hundreds of times the Landauer limit for even a minimal logical operation. For a quantum computer, the control circuitry that carries out the circuit steps uses at least the Landauer limit [26] of energy for each transform. This means that even a quantum computer, as envisioned today, could not carry out more than 2^{128} transforms (gate operations) without running out of energy on earth or exceeding the expected lifetime of the solar system. However, some quantum transforms can do an exponential amount of work in one step, so "impossible" for a quantum computer can be quite different than "impossible" for a classical computer.

3 What is Quantum Computing?

As Niels Bohr said to Wolfgang Pauli and Werner Heisenberg, "Anybody who is not shocked by quantum theory has not understood it." [22].

Quantum phenomena result from the natural structure of matter and energy, and they are as real as anything we encounter in daily life. Yet the fundamental nature of quantum computation is counterintuitive, complex, and deeply confusing at times. The following sections are meant as an informal, gentle, and shallow guide to quantum computation and its embodiment in physical apparatus. It is not an instruction manual, not a textbook, it might not even be accurate in all respects. The interested reader may find it to be useful preparation for discussions with technologists in the various quantum disciplines.

3.1 A High-Level Summary of Quantum Computation

The abstract model for classical computation, the stuff that runs on our computers now, assumes that data is encoded in binary bits. A bit has a single, readable value that lasts at least as long as power is applied to it. Most bits are both readable and writable. A fixed set of electronic circuits can operate on the data and carry out an instruction. An addressable memory holds the data and the instructions. The computer fetches an instruction and applies it to the data.

Quantum computation is radically different. A quantum bit is an object so small and with so little energy that its behavior is governed entirely by the strange world of quantum mechanics and discrete energy units. A quantum operation can involve an exponentially large number of inputs simultaneously and in the same physical space. Accepting this may seem like believing seven impossible things before breakfast, but it is, in fact, well-founded physics that was developed early in the twentieth century.

A quantum "computer" is really more like a tree of logic components than a classical computer processor. It is not a stored program computer with memory. There are no loops, and all results are reversible, i.e., the input state can be recovered from the output state by reversing the order of operations. Algorithms are written as a sequence of reversible operations, and the operations are applied by an external control system.

The essential elementary component of a quantum computer is a qubit. Qubits are cold captives, spinning and singing tunes that only other qubits can hear, until they collapse in a single chord. Each tiny prisoner can be set to ringing its spherical music. Every qubit is great, it can contain multitudes, and together the multitudes are locked in an expanse of logical possibilities.

Qubits are physical entities that are restricted to specific energy states. There is no continuous transition from one state to another, the qubit is either in one or the other state when measured. But when it is not measured or pelted by the outside environment, it can be in a superposition of states. The superposition is not an artificial construct, it is not a mathematical abstraction. Superposition is reality.

Classical computing relies on a "bit" that is implemented by a transistor and has two measurable electronic levels that are assigned the names "0" and "1". A qubit can also can be measured with the result being the 0 and 1 states, but in general that measured value cannot be assigned to the qubit until it is measured in that basis. More generally the qubit state is a probabilistic linear combination of the physical states. Reading a qubit is like flipping a coin and asking while it is in the air whether it is heads or tails: a perfectly legitimate and useful answer is "yes, I'm either heads or tails".

Qubits are controlled by quantum circuits and the circuits are built from gates. A quantum gate is like a simple classical logic gate, e.g. NOT, NAND, etc., except that a quantum gate is not built from static subcomponents but from precisely controlled changes in the electromagnetic field. A quantum circuit is like a classical computer circuit, but it is not built from wired gates but from a sequence of quantum gate operations. Each gate operation takes some amount of time, each circuit uses an amount of time equal to the summation of its gate times.

Quantum circuits are subject to limitations that classical circuits are not. A quantum circuit cannot destroy information, nor can it create independent copies of a state. Circuits cannot have loops. Most importantly, a quantum circuit must be reversible. This means that in one way or another, the relationship between the inputs and outputs must be preserved in the circuit itself.

A quantum circuit takes one or more qubits as input and produces a state(s) that is a function of the input's probabilistic states. The operation is reversible, which means that all the input information is available after the operation. This means that the qubit dependencies that are used in a circuit must be bound into the computation. In the parallel analogy, it is as though each of the parallel computers holds a complete trace of its computation.

When used for computation, qubits are fickle and evasive. Although a qubit can express a complicated superposition of states, the details of that superposition can only be inferred, not read. If you look at (i.e., measure) a qubit before you have finished completing all the operations of an algorithm, the qubit's complicated state will "collapse" into a simple "0" or "1" and will thus lose the result of the computation. Consequently, debugging a quantum algorithm cannot be an interactive process.

Qubits are fickle because they are error prone. The are perturbed by small variations in environmental energy and through sheer randomness. Compared to classical semiconductor bits, qubits are frustratingly unreliable. Fortunately, it is possible to build quantum circuits that cleverly correct the unwanted variations in qubits. The circuit provides error correction so that an aggregate of qubits (each one unreliable) can operate as a single, reliable "logical" qubit. Building error correction gates is one of the major challenges facing quantum computer engineers.

There is one loophole in the fickle nature of qubits that is useful for error correction. If there a point in the circuit evaluation where nothing further depends on the qubit value, and if it is not entangled with anything that will be used later, then the qubit can be measured without disrupting the computation. Qubits with this property are called ancilla bits, and they are very useful for error correction and for coercing an "answer" out of a quantum circuit. However, once an ancilla bit is measured, it cannot be reused, so they become an important part of the resource usage for a computation.

Ordinary bits can be operated on by circuits built from simple electronic gates. Quantum systems also build circuits from gates, but the gates are different. A gate cannot "lose" information, so there is is no "fan in" and no copying (this is subtle; copies can be made, but they can't be used independently of one another). The restrictions stem from a basic principle of quantum computation: operations are reversible. This means that no information can be duplicated, and no information can be lost. The quantum computer's job is to hold the qubits and carry out "circuits" by responding to the external stimuli that change the physical states of the qubits. The stimuli are fluctuations in the electromagnetic field and the passage of time itself.

But the magic has two further principles that complicate computation. Although all the results are computed simultaneously, the qubits cannot be examined until the computation is complete. It is as though there is a genie watching you, and if you peek before the wish is granted, then the genie punishes the wisher by taking away the answer. Second, having all the answers at once is of little use unless you can find the one that matches your success criteria. If you try to examine the results, you get only one, and the others vanish. In the model of the classical parallel computer, it is as though the computer that has the answer cannot communicate — it cannot print the result or write it to a network server. Quantum algorithms must do their computation and leave "breadcrumbs" that hold the answer.

The breadcrumb trick lies in using an alternative dimension of the quantum state, called the "phase", to annotate the favored states, i.e., those (or the one) that constitute the answer. Because phase does not alter state relationships, it can be measured (i.e., "read") from the quantum computer without destroying the relationships. For quantum search, the answer is the bits of the input that produce the goal value; for factoring, the answer is the bits in the number that counts the distinct values of the powers of a member of an integer exponentiation group.

Because quantum bits are error prone while at rest and while being operated on by circuits, any computation of more than a few steps has to involve additional quantum circuitry for error correction. This is one of the largest computations that a QC performs. An error correction circuit may be a few levels deep and involve a thousand (or many thousands) of supporting qubits. The circuits take inputs from some of the qubits and apply transforms to move them from invalid states to valid ones.

When the possible states for a collection of qubits are mutually exclusive, we say the qubits are “entangled”. It is a property that, once established, is preserved by any quantum circuit (mathematically, a unitary operator). A quantum algorithm will entangle the qubits, but not necessarily into one massive clump. The degree of entanglement resulting from a computation is an interesting question of some complexity, but it simply a side effect of the principles of quantum computation. Algorithmists and circuit designers do not need to calculate it *a priori*.

The operations that govern qubit state changes can be described with mathematical elegance by linear algebra’s unitary transforms (which can always be expressed as Hermitian operators). There is a transform called a Toffoli gate that is particularly useful because designers can use it and another simple gate to calculate anything that a reversible classical logic circuit can do. A Toffoli gate uses three input qubits. There are two control inputs, and a third input called the target. The target bit is XOR’d with the logical AND of the two control bits. I.e., the target bit is changed if both input bits are set.

Algorithms for quantum computing are often analyzed in terms of the number of qubits and the number of Toffoli gates that they need. The speed of a Toffoli gate depends on the efficiency of the quantum error correction and the speed of the elementary qubit gate operations. This paper uses very rough estimates for these quantities so that the running time of various cryptographically relevant algorithms can be compared.

One caveat about quantum computing: not all algorithms are faster in QC. Some things will have exponential speedup, some will have quadratic speedup, some will show no improvement, and some are slower. Generally, things that involve simple search and period finding will be faster to *much faster*. Cryptographic algorithms that cannot be attacked by either of these are the subject of NIST’s PQC standardization effort. The proposed algorithms have the property that the search space is so large that even with quantum improvements, the number of steps in the attack algorithm would be impossible to carry out in less than astronomical time. Even though the attack is infeasible, the time to carry out the essential operations of signature generation and verification is within the capacity of an ordinary computer (though it would be stressful for embedded devices [33]).

3.2 Overview of Quantum Computing Device Technology

It is difficult to describe what a quantum computer is in a physical sense because it so unlike any other kind of computer. Many descriptions of the basic unit, the qubit, describe it as an oscillator in an electromagnetic field. The energy of the oscillator is what makes it different from our usual electronics devices. Very small objects, things like atoms and electrons, have energy, but only in discrete chunks, "quanta". Increasing the energy of a very small object can only happen when the amount added is large enough to move it to a new level, a level that is determined by fundamental physical properties. Similarly, energy is only decremented in discrete amounts. The energies involved are quite small, so the qubits need protection from thermal noise, and refrigeration to extremely low temperatures is the solution. The most evident part of a quantum device is the large cryostat (a cylinder filled with liquid helium).

There are many things in nature that can act as qubits; some are easier to control than others. The physical structures that cause something to be subject to the principles of quantum mechanics are varied, complicated, and imperceptible in ordinary daily life. Atoms, electrons, atomic nuclei, etc. can be coaxed into behaving as qubits in a laboratory. Two major technologies dominate quantum computing research today: superconducting qubits and trapped ion qubits. Trapped ions systems have more stability than superconducting systems, but superconducting systems offer greater speed. Spin qubits in semiconductors also appear to have near-term feasibility. There are many other possible approaches, some purely theoretical [8], some only lightly explored [11].

Despite the differences in physical realization, all qubits have fundamental properties that allow them to be acted on by transforms, put into superposition states, coupled with other qubits, and read.

- Qubit Properties

  The physical apparatus for a qubit must have some way to set the qubit into one of its known and measurable states, to perform transforms on the qubit state, to perform transforms involving two or more qubits, and to evaluate the state. Microwaves are the usual signals for causing the physical changes that are essential to using qubits. An excellent description of recent methodology can be found in [9].

  How is superposition done? A qubit state is based on its lowest two energy states, which are fundamental physical properties of the implementation technology. In a superposition, the qubit has a probability distribution between the two ground states. That distribution can only be measured probabilistically over several experiments; it cannot be directly observed. The exact mechanism for creating a superposition state depends on the implementation technology, but the general physical principle is that the qubit has a "resonant frequency" that is derived from its physical properties. An energy pulse of that frequency for a particular amount of time directed at the qubit will evolve the qubit state to a new state.

For the example, the transform that moves a qubit from a ground state into a superposition where both the 0 and 1 states are equiprobable is called a Hadamard gate. It is algebraically represented as a matrix

$$H = \frac{1}{\sqrt{2}} \begin{vmatrix} 1 & 1 \\ 1 & -1 \end{vmatrix}.$$

An infinite number of other transforms on qubit states are possible.

It isn't always easy for a quantum engineer to turn a matrix into a set of manipulations that have the desired effect and the required fidelity. A simple set of high fidelity gates is desirable, but they might not be sufficient for a particularly stressful algorithm.

- The abstract properties of a qubit. A qubit is often described as a vector on the Bloch sphere. This means that it is a vector of unit length with x and y coordinates that are the sine and cosine of an angle θ measured from the xy plane, and a z coordinate that is the angle of rotation ϕ around the z axis. This last quantity is called the qubit phase.

  Single qubit transforms can move the qubit into different positions on the Bloch sphere. The positions reflect the qubit's state probabilities and its phase.

  A qubit can be related to a physical property through measurement, and the measured value will be one of the base states. The probability of the measured state depends on the measurement basis, though. Any orthonormal vector set can be used for the measurement basis, but unless the qubit state is composed of the same vectors, the measurement will not be related to the qubit state.

  Note that in general a qubit does not "have" any measured value until it is measured. In a superimposition of states, the qubit exists in all the superposition states at the same time. Measurement forces the qubit to commit to one state, and the others disappear.

- How are two-qubit gates constructed? Two qubits can combined in a way that makes their joint state interdependent. To do this, the qubits must be electromagnetically coupled so that they can influence one another. The coupling involves capacitances so that the combined system is something like an oscillator. The period of the oscillator is an important factor in determining the time to perform a unitary transform. Carefully tuned (the tuning can be fixed or adjustable for a transform) frequency pulses cause the two qubit states to be locked into a joint distribution. Magically, this combined state will persist when the coupling is deactivated.

For a description of two-qubit gate design for superconducting qubits, see [18].

For classical electronic design, the workhorse gates are things like NAND and NOT. For quantum devices the workhorses are the single qubit gates and the controlled NOT gate called “CNOT”. The CNOT gate is very similar to an XOR. It takes two qubits, called target and control, as inputs. The control bit determines if the state of the target bit will pass through unchanged or be negated. Unlike a classical gate, the CNOT does not destroy the original state of the target bit. It essentially carries the information “if the control bit was 0, the target bit is X; if the control bit was 1, then the target bit is not(X)”. That statement is the state of the two bit system after CNOT is executed on the input bits. This construction can be generalized as a conditional control on any operation, i.e. “if A then do B”.

Although single qubit gates combined with CNOT gates is enough to carry out any quantum transform, there is a time cost for each combination. Depending on the qubit technology, and perhaps on the algorithm, additional gate types will be useful because they will be much faster than synthesizing specific gates from a small universal set. The circuit controller can have sets of complex gate combinations to serve as quantum circuit subroutines. Gates the operate on two or more qubits at a time are especially likely to need special purpose circuits. This is analogous to custom circuits in the classical world for use in ALUs, e.g. carry look-ahead adders.

- Coupling. In order to perform a transform on two or more qubits, they must have electromagnetic coupling between them, and this is typically done with small metal layers. Once the transform is complete, the coupling can be deactivated, and the qubits will retain their new state. For a complicated algorithm, like multiplication, each qubit will need to be paired with each of the other qubits in the circuit at some point or another. It is infeasible to create a complete wiring graph on thousands of qubits, so the qubits will have to be "moved" to the coupling junctures as needed by the algorithm. This is possible because qubits can be "swapped" without running afoul of the laws of quantum mechanics. Nonetheless, the swaps use time, and it is worthwhile to design a topology of qubits that minimizes their need. This is another point where algorithmists and quantum engineers may need to collaborate in order to guarantee success.

The quantized oscillating objects are what make up the basic unit of quantum computation: the qubit. Of course, all of the universe is made up of quantized particles, but when they are controlled and confined in laboratory devices, they can be used for computing. For ordinary computing devices, circuits control how each bit in the computer is set to a specific value, 0 or 1. For qubits, the situation is more complicated because the bits have complicated values (superposition) that are co-dependent on other qubits (entanglement).

- How is measurement done? Measurement of qubits is an interaction of the qubit with the environment that causes an observable change: the emission of a photon, for example. The change is determined by the physical properties of the qubit implementation, but it will tie the qubit to one of its physical states. The measurement is done in a "basis" which is a set of mutually exclusive state combinations. Unless the qubit has been put into the measurement basis, the outcome will be random.

For a good description of Google's superconducting quantum computer, see [23].

3.3 What are the Challenges Facing Quantum Computer Development?

Superconducting QC technology today uses microwave wavelengths to drive the circuits because high-precision microwave generators are readily available. This means that the lower limit to a single qubit transform is in the gigahertz range. In theory, speeds up to terahertz are possible, but the engineering considerations have not been explored.

The expensive resources in quantum computing are the number of qubits, the error correction complexity, the amount of qubit coupling, the time to perform a set of unitary transforms, and the cooling during operation of the circuits. All of these depend heavily on the physics of the device.

Qubits are expensive resources because they are very small and have to be refrigerated and coupled and connected to microwave signals. Minimizing the number of qubits is a key goal of quantum algorithmists. The error prone nature of quantum devices requires devoting a lot of quantum resources to creating reliable qubits that can carry out an algorithm. An error-corrected qubit is usually called a “logical” qubit. This leads to expressing the time and resources for a quantum algorithm in terms of logical qubits. The error correction expansion is expressed in terms of a multiplier for the number of supporting physical qubits and the time to carry out the error correction for a Toffoli gate.

Qubit gates are implemented by changes in the electromagnetic field. Single qubit transforms are fairly straightforward, and there is a trade-off between the time to perform the transform and the error probability. Gates that use two qubits are slower because the qubits must be electromagnetically coupled while the transform is applied, and the physics of the coupling introduces an oscillatory factor that limits the speed of change. Some physical technologies favor some kinds of transforms over others. For example, a phase gate is particularly easy for superconducting transmon bits. This means that quantum engineers may need to work with the algorithm designer to find the best set of gates for implementing an algorithm. The quantum world is not as strictly layered as the classical world!

The time for carrying out a quantum gate operation depends on the operation and the physical properties of the qubits. To estimate the time for running a quantum algorithm we need to ask the quantum engineers how each of the required basic transforms is constructed. The low-level universal building block gates are the Hadamard and CNOT gates (see [39]), but circuit designs that are limited to these will probably not be optimal in terms of time or fidelity.

Error correction is the subject of ongoing design and evaluation in quantum computing research. The computation time for a quantum computer will be entirely dominated by error correction. Because there are significant gate errors inherent in quantum devices, achieving a reliable qubit will require somewhere between one thousand and one million additional qubits with many connection gates in a two-tier logical configuration.

The circuits for error correction depend on the probability of error in a qubit state, which depends on the quality of the qubit implementation and the coupling of gates and the topology of the qubit layout. Minimizing the error correction circuits is a key goal of quantum computer designers.

The speed of a circuit, such as a Toffoli gate, depends on the inherent qubit coupling parameters and the speed at which the transforms can be driven. Usually higher speeds result in higher error probabilities.

Any transform will introduce some amount of heat from the control circuitry. Heat build-up can limit the number of operations that can be performed in a given time, necessitating cooling cycles.

The number of quantum gates largely determines the running time of an algorithm. Although Toffoli gates make it possible to translate any classical reversible computer circuit into a quantum circuit, a straightforward translation might have a lengthy gate sequence and be greatly inefficient. Quantum engineers, physicists, mathematicians, and algorithmists, working together, might develop maximally efficient quantum circuits that optimize the core operations for particular algorithms, like Shor's, with customized designs for modular addition and multiplication [19].

Because qubits are inherently error prone, they must be built with error correcting circuits as part of their usage. If the qubit technology reaches a high enough "fidelity" (the opposite of error), then it is mathematically possible to build a hierarchy of quantum error correction circuits that will move the system from an error state to a corrected state. A good discussion of circuits for achieving this using surface codes can be found in [17]. Error correction circuits themselves, however, need a substantial number of qubits. Although the final say on this depends on further research, it seems that about 1000 qubits are needed to implement one circuit that can act as a reliable, usable logical qubit. IBM aims to demonstrate a reliable qubit in 2023. Should this be successful, the company expects to manufacture logical qubit chips that can be connected into quantum registers.

Operating on a single qubit is relatively easy, but to carry out useful computations, at least two or three qubits must be involved. In classical computing, ordinary circuits can easily be designed to perform, for example, 32-bit addition. For quantum computation, in order to get the benefits of massive parallelism, the qubits must be coupled together in an electromagnetic field, and they will respond to energy pulses much as if they were coupled oscillators. As a result, the time for a quantum state change is limited by the coupling parameters. In general, a quantum gate is expected to be 10 to 1000 times slower than today's fastest processor clock speeds, and quantum circuits with error correction will be 10,000 to 1,000,000 times slower than a typical classical computer instruction.

3.4 The Mathematics of Quantum Circuits

Classical computing circuits follow the rules of Boolean logic formulas, but quantum circuits adhere to linear algebra and unitary transforms. The transforms describe how a qubit or qubits can move from one superposition state to another. Ordinary electronic circuits can carry out any finite logic transform, but for quantum circuits, conservation of energy and momentum for the quantum states limit the transforms to those that can be described by unitary matrices. The matrix eigenvalues and eigenvectors are related to the quantum system energies, and the mathematics and the physics intertwine in a elegant, if sometimes opaque, set of relationships.

Finding the transform that can carry out a given function is not always easy, and some functions may require a very complicated transform. There are some elegant theorems that show that any single qubit transform can be constructed from a few simple 2-by-2 matrices (e.g., the Hadamard (H), phase(S), and $\pi/8$ (T) gates). Nonetheless, a particular construction, say for a small phase angle, might end up using a lot of gates, and therefore a lot of time. A quantum engineer must try to balance the gates that have the best fidelity with the time requirements of the algorithm.

For 2-qubit gates, there are also decompositions into elementary transforms, but the qubits for the operation must be electromagnetically coupled. Without the coupling, the computation cannot express dependencies between bits. When the coupling is in place, unitary transforms on qubits can be carried out through changes in the field. There are an infinite number of transforms, but any transform can be decomposed into a combination of (for example), CNOT and Toffoli gates. Quantum algorithms are often expressed using these gates because the Toffoli gate can express any reversible logic circuit. As with single qubit transforms, the details of gate efficiency will depend on the fundamental physics of the quantum device. Google is developing novel gate design methods to facilitate design variability [18] [30].

The composition of simple gates into useful circuits is a sort of machine language for quantum computing. Unlike classical computing, where a circuit can be built of any combination of logical expressions, a quantum circuit is a unitary transformation that acts on a vector of state superpositions and produces a new set of state vectors as output.

For the cryptographic algorithms of interest, the algorithm developers generally express their resource estimates in terms of a gate that is universal for reversible logic (e.g., Toffoli gates). The number of gates serves as a rough measure of the running time of the quantum algorithm, and the count helps when comparing the efficiency of different algorithms. Algorithm developers need to express their entire computation in terms of these basic gates. For cryptographic computation, they will have to implement a complete set of arithmetic operations. Developing these quantum libraries will require using many of the same computing tricks that a classical algorithm would.

To be useful on a quantum computer, the algorithm must exhibit substantial reduction in resources and time for the parameters of interest in cryptographic protocols in use today. That reduction may seem obvious given the quantum computer's ability to work on 2^n inputs in parallel, but if the quantum gates cannot do the arithmetic efficiently, the quantum advantage may be greatly weakened.

A good introduction to quantum computing principles can be found in Chapter 2 of [39]. The Qisket language [5] is a quantum circuit language and it has a graphical interface for using quantum gates and displaying their effects on state vectors, composing gates, and simulating them using parameters from IBM quantum devices.

4 Cryptographic Security Algorithms

Privacy and authentication are the compelling use cases for cryptography on the Internet. Privacy concerns long-term communication secrecy, and authentication is about the digital artifacts that establish trust in communication (identity, authorization, etc.).

The important algorithms for today's Internet security needs are those for hashing, encryption, key exchange, and digital signatures.

4.1 Privacy Considerations

The early Internet was able to thrive in spite of lack of encryption on communication lines. Once financial transactions started becoming part of the online environment, encryption was a necessity. In the 1970s, software encryption was too slow for communication use, and there were some efforts to provide hardware devices, such as the STU-III phone. These proved awkward to use, and it was clear that it would be best if the parties involved in communication could do their own encryption on their computers and achieve end-to-end encryption at will. By the 1990s it became possible to run the Data Encryption Standard (DES) on ordinary computers at communication line speeds, but by then DES was considered insecure because of its very short key length (56 bits).

A lot of work went into trying to find encryption methods that walked a fine line between security (generally measured by key length and "effort to break") and usability (generally measured by the time to encrypt 64 or 128 bits of data). The final outcome was a jump to the Advanced Encryption Standard (AES) and its 128 bit key length.

Today AES is supported in custom hardware included on Intel processor chips, and there is no need to find any balance. AES is fast in software, and the 128 bit key length means data will remain safe for many hundreds of years or more, even if quantum computing becomes a reality. In fact, most Internet communication today uses the AES option of a 256 bit key, which is somewhat slower than the 128 bit option, but not enough to bother anyone, and it somehow gives peace of mind in being astronomically more secure.

4.2 Hash functions

Hash functions are integral to common forms of password validation, some key exchange algorithms, and DSA signatures. The SHA-2 family of hash functions have had a great deal of scrutiny, and they are recommended for Internet usage. The functions are secure against even quantum computation (see Table 1). Our analysis is based on estimates of how much time would be needed, but even beyond that, an attack would need a huge amount of quantum memory and is unlikely to be realized in any quantum scenario.

SHA hash, time to break with quantum computing

#bits	l-gates per hash	l-gates	l-qubits memory	time* Kyears	physical memory qubits	physical time KYears
160	25K	$2.7 * 10^{20}$	$7 * 10^{18}$	$8.5 * 10^{3}$	$2 * 10^{22}$	$1.0 * 10^{5}$
224	45K	$1.4 * 10^{27}$	$2 * 10^{25}$	$4.4 * 10^{10}$	$5 * 10^{28}$	$5.3 * 10^{11}$
256	45K	$2.3 * 10^{30}$	$4 * 10^{28}$	$7.2 * 10^{13}$	$1 * 10^{32}$	$8.6 * 10^{14}$
384	110K	$3.6 * 10^{43}$	$5 * 10^{41}$	$1.1 * 10^{27}$	$2 * 10^{45}$	$1.3 * 10^{28}$
512	110K	$2.5 * 10^{56}$	$3 * 10^{54}$	$7.8 * 10^{39}$	$1 * 10^{58}$	$9.4 * 10^{40}$

Table 1: #bits is the output block size
The 160 bit size is SHA-1; all others are SHA-2.
* this is the unscaled time, 1 μ sec per gate
Based on [13] with scaling based on [19].
memory = $\sqrt[3]{2^{\#bits}} * (\#bits + \#inputbits)$

4.3 Symmetric Encryption

Symmetric encryption scrambles the bits of a message according to a function that uses a key of 128 bits or more. Anyone who knows the key can easily extract the data. Anyone who does not have the key will have to try guessing it by running the encryption algorithm and checking the result against some criteria known to match the data. Modern encryption algorithms, like AES, have all the necessary long-term security needed for any purpose.

4.4 Key exchange

The classic Diffie-Hellman algorithm can be implemented over any cyclic group for which the discrete logarithm problem is hard. The two parties to the key exchange (Alice and Bob) agree on a symmetric encryption key by each raising a pre-determined group generator to their own secretly chosen power. The values are exchanged, and each party raises the received value to their own secret power. The resulting values are equal, and the key for encryption can be derived by well-known methods that usually involve hashing.

The calculations can be carried out efficiently over either a modular exponentiation group or an elliptic curve group. The number of bits in the calculations in the former case must be about 10 times larger than the number of bits in the latter case in order to achieve roughly equal long-term security.

4.5 Public Key Authentication

Public key authentication methods for RSA or DSA are based on entities publishing their public keys, and then using a related private key, to perform a non-reversible function on the hash of message data. The hash is published as the signature on the data. The result of that function mathematically proves that signer knows the private key that shares a mathematical relationship with the public key.

A public key of a few thousand bits is published by the signer. Anyone can validate the signature. A legal document, like Prince Philip's will, might need to have an authentication result that could be validated over the lifetime of the agreement, say 90 years. When an authentication key is "broken", an impostor can pose as the key owner and sign data. The security implications of this vary from minor to serious.

1. If a key is compromised and used to sign bogus information, it is usually detectable because a significant amount of time must pass between publication of the public key and the "break" of the key. If the metadata for the key has an expiration date that has passed, the signed data can be shown to be improperly authenticated.

An interesting illustration of how outdated keys are handled occurred recently. The LetsEncrypt root certificate expired, breaking many older web browsers that did not update root key lists [3]. The expired key had been used to authenticate the website keys, but once it passed its expiration date, the web browsers quite properly indicated to the user that website keys were invalid. This event demonstrated that key lifetimes are an effective way to protect against future forging.

2. If the key owner immediately detects the bogus signature by monitoring areas that use his signed data, he can take steps to override the publication of the signed data and also publish a new key that is authenticated by a trusted party. Critically important data should be signed by multiple trusted parties.

3. There is always the possibility of an out-of-the-blue attack that breaks many keys in a hierarchy and undermines the whole authentication system. This might happen because of a mathematical breakthrough in factoring or because a new computing paradigm emerges without warning (such as quantum computing). That could wreak havoc if keys were quickly broken and forgeries were running rampant. The remedy for this is to periodically re-sign the data with a new key that is validated by trusted parties who can attest to the signer's integrity.

4.6 Factoring and Discrete Logarithms, Finite Groups

The RSA algorithm for signatures depends on the difficulty of factoring the public key. Factoring effort is subexponential (and superpolynomial) in difficulty. Attacking Diffie-Hellman key exchange over multiplicative groups has a similar level of difficulty because discrete logarithms can be attacked in an almost identical way.

The effort for factoring a number Nscales as

$$e^{1.923\sqrt[3]{n*(log n)^2}}$$

where n is the natural logarithm (base e) of N.

The formulas for effort depend on constant terms that embody the fixed effort of basic arithmetic and matrix operations. Those constant terms are significant in the range of interest (from 1024 to 4096 bits). For example, improvements in the number field sieve (NFS) factoring method [28] on modern processors show that the effort of solving a discrete logarithm problem for a 795 bit number uses only about 3 times as much effort as factoring a number of the same length [12]. Although discrete logarithms use somewhat more complicated operations in the final step of the process (the row reduction), the relation gathering step, which is the most time-consuming part of the process, is equally fast for both factoring and discrete logarithms.

4.7 Discrete logarithms for DSA style signatures

The DSA method for authentication and the DH method for key exchange both depend on the difficulty of discrete logarithms to ensure security. Discrete logarithms are the inverse to exponentiation in a group. If k is the value of g^x for a specified group generator g, then x is the discrete log of k. If the group is very large, then the discrete logarithm problem is very difficult.

Two kinds of algorithms can be implemented. The underlying mathematical structure can be integer modular exponentiation on subgroups embedded in a large multiplicative group or an elliptic curve group.

The two security algorithms of concern in this class which are of particular Internet importance are the Diffie-Hellman key exchange, which is part of TLS and IKE, and also DSA signatures, which are embodied in NIST documents [1] and the IETF XML signature documents. Some noteworthy dependencies are HTTPS [41], TLS [40] and DNSSEC [43] which are together the foundation for the security of web browsing. For VPNs, we note that key exchange protocol IKEv2 [25] depends on the security of Diffie-Hellman key exchange.

Discrete logarithms over modular exponentiation groups are subexponential in difficulty, and they are only slightly harder than factoring [12].

The security of the algorithms depends on the number of bits in the keys and the number of operations required to "undo" the key security. For classical computing, factoring algorithms have subexponential difficulty due to the efficiency of the Number Field Sieve. Though "subexponential" may sound like factoring is easy, the ratio of effort goes up steeply with respect to the number of bits in the target number. A number with 2000 or more bits is impossibly far out of range for classical computers.

Note that the actual time used depends on the type of processor cores used in the calculation. Recent factorizations use less time than one might predict by extrapolating from factorizations of several years ago. Even with the same clock speed, modern processors are noticeably better at factoring than their predecessors.

What does it mean to have safe cryptography at the present time? We can measure the amount of effort that an adversary would have to devote to "breaking" a cryptographic key in terms of time, energy, or money. Energy is a more consistent measure because time depends on the number of processors devoted to the task. Multicore technology is producing more processors for less money, and we cannot ignore that progress in assessing future risk. The same can be said for specialized hardware like GPUs.

Most algorithms in use today for website security are probably safe forever in that it would take a truly infeasible amount of effort to break even a single instance of usage. In the past, it was necessary to balance the cost of using cryptography against the risk of it being successfully attacked, and people would talk about 10 or 20 or 100 year security. The computing advantage today is firmly in the hands of the protectors, not the breakers, so we can talk about "secure unless quantum computing becomes a reality".

The caveat about security arises because quantum computers have a much lower ratio of effort between attacks on one key size and attacks on a larger one. Rather than being subexponential, it is n^3. That means that if a quantum computer could factor a 1000 bit number (which is just barely feasible today with a million classical computer cores and 4 months) in N days, then it could factor a 2000 bit number with only about $8N$ days. This ratio makes life too easy for the attackers.

The bottom line is that the algorithms that underlie much of Internet security, and are highly secure with respect to classical computing, are fragile against quantum computing.

For Internet security, we ask, "When can quantum circuits be used to undermine the security of the important cryptographic algorithms that are used heavily in Internet communication?" For reasons that will be justified in the next section, we call this "quantum factoring superiority" (QFS) or "quantum discrete logarithm superiority." At the current time, there are no convincing demonstrations of quantum factoring other than toy examples. QFS cannot be approached until a time when quantum computing is a mature technology.

4.8 Attacks on Keys and Groups

Some cryptographic attacks do more damage than others. Some ruin only one user's key, others ruin the choice of the group for the protocol. These are vastly different risks, as delineated here:

- Breaking an RSA key endangers only that key.
- Breaking Diffie-Hellman over a multiplicative group of a finite field of prime order ruins that prime for all further use.

- Breaking Diffie-Hellman over an elliptic curve group breaks only one private key.
- Breaking DSA over a hidden multiplicative group subgroup breaks that system of group parameters completely if the break is for discrete logarithms modulo the large prime.
- Breaking DSA over an elliptic curve group breaks only that one public key.

For all these systems, protocols in use today use structures that are "impossible" to attack in classical computing. Diffie-Hellman key exchange, for example, when using 2048 bit multiplicative groups, is completely safe for classical computing. As we will see, that picture changes dramatically for quantum computing.

4.9 By the Numbers: Current number theory computation results

Some algorithms need thousands of bits in their parameters for security against classical computing, others need hundreds. This section shows the effort need to "break" the various systems for various parameter sizes.

The best algorithm today for factoring integers is the Number Field Sieve (NFS). The security of the RSA signature method depends entirely on the difficulty of factoring. Implementations of NFS continually improve at an incremental but steady rate. The algorithm is parallelizable and is successful at factoring numbers with close to 900 bits on supercomputers today. Some recent work is summarized in the first columns of Table 2; also shown are extrapolated times to break larger moduli.

Note that the RSA1024 number is interesting because it suggests that a supercomputer the size of the Fukagu [4] or Sunway [34] could complete the factorization in under a few months. The US is rumored to be close to unveiling a machine of greater power this year.

The Diffie-Hellman key exchange method, when implemented over modular exponentiation groups (multiplicative groups), has roughly the same security as factoring integers of similar size. Although the attack needs more memory to store the intermediate results (relations), this does not greatly affect the classical processing time as compared to factoring numbers of similar size.

Number theoretic security algorithms other than RSA have the option of using arithmetic defined over group structures that are not the usual mul-

Factoring times, classical and quantum

number name	bits	classical time KCYr	qubits logical	qubits physical	gates Toff+T/2	QC time hours
RSA 240	797	1	2406	8M	150M	< 1
DLOG* 240	797	3.2	2406	8M+	150M	1
RSA 250	830	2.7	2506	8M	190M	< 1
RSA 1024	1024	500	3092	10M	400M	1.5
RSA 2048	2048	$6 * 10^{11}$	6189	20M	2.7G	7
RSA 3072	3072	$2.2 * 10^{18}$	9287	40M	9.9G	13
RSA 4096	4096	$5 * 10^{23}$	12386	50M	23G	20

Table 2: KCYr = kilocore year
Classical computing, time to complete on processor: Intel Xeon Gold 6130 at 2.10 GHz; 16 physical cores.
*Discrete logarithm (not factoring, but greatly similar)
RSA240, RSA250, DLOG240 are from [12].
Extrapolation formula:
(N = log(2) * number of bits), $e^{1.923 \sqrt[3]{N*(logN)^2}}$
scaled up from RSA250.
Quantum estimates based on [19], Figure 1 and Table 1.

DSA signatures, time to break with classical computer

Bits in Hash and Q	P bits	Hash collision	discrete log Q	discrete log P
160*	1024	6.5	$9 * 10^5$	$1.5 * 10^3$
224	2048	$4 * 10^{15}$	$4 * 10^{15}$	$1.8 * 10^{12}$
256	2048	$3 * 10^{20}$	$2.5 * 10^{20}$	$1.8 * 10^{12}$
256	3072	$3 * 10^{20}$	$2.5 * 10^{20}$	$6.6 * 10^{18}$

Table 3: Times are in kilo-core years. Time to complete on processor core with with 2.1GHz speed. Estimating 50 cycles per modular multiplication. *SHA-1 hash, broken by [45]; other times are for SHA-2.

Elliptic curve DSA signatures, time to break with classical computer

bits	words	cy/mul	*8	add/sec	$2^{N/2}$	secs	KCYr
224	4	28	224	9.4M	$5.2 * 10^{33}$	$5.5 * 10^{26}$	$1.8 * 10^{16}$
256	4	28	224	9.4M	$3.4 * 10^{38}$	$3.6 * 10^{31}$	$1.2 * 10^{21}$
384	6	60	480	4.4M	$6.3 * 10^{57}$	$1.4 * 10^{51}$	$4.5 * 10^{40}$
521	9	135	1080	2.0M	$2.6 * 10^{78}$	$1.3 * 10^{72}$	$4.2 * 10^{61}$

Table 4: Time to complete on processor with 2.1GHz cycle time.
bits = # bits in the prime P;
words = # 64bit words to hold values mod P;
cy/mult = # of cycles to multiply two values mod P, and reduce the product back to in-range for mod P (assumes NIST primes with fast reduction are used);
*8 #cycles for 8 multiplications
or adding two elliptic curve points, including arithmetic to convert to affine representation to identify equal points. This uses the Montgomery reciprocal trick.
add/sec = # of EC point additions per second
$2^{N/2}$ = search time to solve the elliptic-curve discrete log problem
secs = time (in seconds) to solve the problem
KCYr = time (in Kcore-years) to solve the problem.

tiplicative groups. The significant examples are elliptic curve point addition groups and multiplicative groups "hidden" in much larger multiplicative groups. These can both be attacked using discrete logarithm approaches in which the security is measured by the square root of the size of the mathematical group. Because the square root cost dominates the subexponential cost of factoring, small moduli yield the same security margin as do multiplicative group structures.

For example, RSA with 2048 bits and ECC with 192 bits have about the same level of security. RSA moduli of 2048 bits or higher are impossible to break with classical computing, but 1024 bits is within reach today. Discrete logarithms using 1024 bits will be attackable within a few years. In contrast, elliptic curve groups typically used today have over 200 bits, and the security of these parameters is conservatively estimated at one billion years.

The classical computer times for breaking the DSA signature algorithm for multiplicative groups are listed in Table 3 and elliptic curves times in Table 4. In all cases the processor cycle time is assumed to be 2.1GHz.

The tables show, in some detail, that current Internet cryptographic security algorithms are deployed with a very large safety margin with respect to classical computing. Spoiler alert: the quantum factoring estimates are included in Table 2. The quantum information is discussed in more detail in section 5.3.

5 Why is Internet Security Susceptible to QC Attacks?

Quantum computations for factoring and discrete logarithm computation are expected to be overwhelmingly faster than classical computing. For any size of modulus in use today for RSA or DSA signatures, or for Diffie-Hellman key exchange, a sufficiently powerful quantum computer could find a "break" in at most a few years, and possibly in a day or less. By powerful, we mean that it has several thousand error-corrected qubits and can execute general purpose gates, particularly Toffoli gates or equivalently general gates, in substantially sub-millisecond times.

Keep in mind that quantum computers are not necessarily "faster" than classical computers. The circuits that are being built today to carry out a simple transform are slower than similar classical circuits by a few orders of magnitude. Their advantage lies in being able to manipulate much more information is a single operation (transform) than a classical computer can do in a single instruction. For the purposes of this paper, we assume a very optimistic time of 12μsec per error-corrected two-qubit gate. Although this is not a fundamental limit (which could possibly be a thousand times faster), it does represent the limits of equipment being developed for demonstrations today.

Quantum computers to date are slower than classical computers, and this appears to be an inherent physical limitation. The electromagnetic coupling that allows two qubits to become entangled works best when the coupling is slower. In general, using a lower coupling constant to increase the gate speed will lower the fidelity. If the fidelity is too low, then error correction is not possible. The best guess at this time is that even the best quantum computer gates speeds will lag classical computer instruction speeds by a factor of 1000 (long term) to a million (over the next several years).

We should note that there is a proposal to "save" RSA from quantum computing by using an extremely large public key [10], one so large that a quantum computer would need an impossibly large number of operations to factor it. It suffers from the problem of needing far too many classical operations for the signing and verification operations. This impracticality quantifies the problem of pitting classical operations against quantum operations.

5.1 Symmetric key search and Hash Collisions

With a little bit of knowledge about the data that was encrypted, a quantum computer can search for a key that decrypts the recorded ciphertext and matches the description. The search is faster than can be done with classical computing, but probably not good enough to cause consternation, even for people who need to protect secrets for a good long time.

The usual key size for encryption today is 256 bits. (Because that is overkill by any reasonable measure, we have only analyzed quantum attacks on 128 bit keys. As we will see, that key size is safe for all uses.) Because Grover's quantum search algorithm [39] reduces the effective effort from 2^{128} to 2^{64}, and because a classical computer could search a 64 bit key space in less than a month, we might expect that a quantum computer could easily "break" a 128 bit key. The quantum circuits, however, compute encryptions using one operation for each bit in the plaintext block [20]. In contrast, classical computers can use one operation per 64 or 128 bits. That processing difference, combined with the slower operation speed of quantum computers, means that the quantum advantage is less dramatic.

A billion quantum computers working on AES128 would take 1000 years to find the key. This is not feasible. The energy cost alone of keeping the computers insulated would be prohibitive. There is no need to use AES256 unless one fears an algorithmic attack (finding mathematical weaknesses in the mixing operations, for example).

Hash functions are part of the security for digital signatures using DSA. If it were easy to find hash collisions, it would be possible to create forgeries by requesting a signature on one message and then using the signature as authentication for a different message. Grover's algorithm can be used to find collisions, and the effort analysis is similar to key search. The effort scales as $\sqrt[3]{2^N}$ where N is the number of bits in the hash [13]. However, Grover's algorithm when used for collision finding requires a very large quantum memory, and designing that opens up a new set of problems.

For factoring and discrete log problems, the number of operations for solving problems that would threaten today's Internet security could be carried out in a day or less on a fully realized quantum computer. Other types of problems, such as searching for the 128-bit key used in a symmetric encryption, would take much longer, hundreds of years, and probably would be infeasible even with the speed-up afforded by massive parallelism.

5.2 Shor's algorithm, Period finding

We describe Shor's algorithm at a high level in order to estimate the quantum machinery (circuits) and steps needed to factor a cryptographically relevant number. Because today's classical computers could factor numbers of 1024 bits in much less than a year (albeit at great expense), we consider numbers that are at least 2048 bits to be in the class of things that are secure against classical computing under all scenarios, but are attackable with quantum computers that may eventually emerge.

There are many detailed descriptions of Shor's algorithm including Shor's 1994 paper [44], Coppersmith's improvement of the quantum Fourier transform [15], a description in the context of the algorithm components [39], and for the non-expert [27] and [16]. What follows here are the major steps of the algorithm and some discussion of the gates necessary for implementing them.

The heart of Shor's algorithm is the use of linear algebra to find cycles in number patterns ("period finding"), and that capability is the basis for integer factorization in polynomial time. In period finding for an integer N, the quantum circuits calculate all powers of a base number r using modular reduction with respect to N. Elementary number theory shows that the powers of $r \pmod N$ will repeat in a cycle. The period of the cycle divides $\phi(N)$ (the usual modulus for RSA is a number that is the product of two large primes p and q, and $\phi(p*q)$ is the product of $(p-1)$ and $(q-1)$), and this is almost always enough information to find a factor of N.

In describing the qubit registers for factoring N, it is convenient to define a quantity L which is $\log_2(N)$ rounded up to the nearest integer.

Period finding requires a lot of arithmetic. Some of the powers of r can be precalculated by a classical computer, but most of the arithmetic must be done by the quantum circuits. The precomputed values are the binary representations of $r^{2^i} \pmod N$ where i runs from 0 to $2L-1$. These form a basis for representing any power of r. The quantum circuit will calculate all the necessary powers of r into one entangled bundle. Then it will tease out the period of $r \pmod N$. Let's call the period s.

The quantum circuit begins with an "exponent register" of $2L$ qubits all in the 0 state. Each bit is then run through a special gate, the Hadamard gate, that puts it into a superposition of the 0 and 1 states. In that superposition, a measurement of a bit will yield either 0 or 1 with equal probability. In aggregate the bits hold all the information that represents the numbers from 0 to $2^{2L}-1$. The amazing thing about quantum states is that this aggregate state is all those numbers, all at once, in a real physical existence.

There is another register of L qubits all initialized to state 0 except for qubit 0 which is initialized to state 1. This will hold the powers of r as they are computed by a sequence of transforms.

The exponent register is then used as the driver for the computation of the 2^{2L} powers of r. The reader may wonder why 2^L powers are not sufficient. The powers will repeat beyond 2^L, so the extra powers seem redundant. Those powers will be useful at the stage of the algorithm where the period of r, which is probably not an exact power of 2, will be estimated from the collection of powers that have equal values. More values mean more accuracy.

Carrying out the modular exponentiation of r uses a unitary transform for each precomputed 2^i power of r. The exponent qubits control the application of the transform. Qubit i is the input for a control bit to the logical circuit: "for the case of state 0, do nothing; for the case of state 1, apply the transform that multiplies r^{2^i} to the result register and reduces it modulo N". The number of gates for implementing the details of modular multiplications, additions, and reductions is immense and is the major challenge facing a quantum factoring engineer.

Because quantum transforms are reversible, all possible powers of r, along with all the dependencies on the qubit superimposed values, are held in the result register. The powers repeat in a period, so we expect to find many instances of each of the unique powers. The number of instances will be about $2^{2L}/s$ where s is the period. What is needed next is a way to count the unique values without collapsing the computation state.

In section 3.1 we mentioned the 3 dimensions of a qubit and the trick of notating the desired answer with a "breadcrumb". The phase angle ψ is the breadcrumb. There is a mind-boggling trick in quantum circuits that can push the phase of a transform into the control bits without changing the result of transform. This is called "phase kickback", and sets up the qubits so that further computation can be done using the qubits to represent frequency domain inputs to the quantum inverse Fourier transform. Each power of r in the result register has a phase angle ψ that is the sum of unique roots of unity. The inverse Fourier transform will produce a distribution with the count of each of the different phase angles.

The quantum Fourier transform requires gates that can rotate the qubit phase angle by specific fractions of π. As noted earlier 3.3, superconducting qubits can be phase rotated by exciting them at their resonant frequency. The rotation angle is linearly related to the duration of the excitation.

Inverse Fourier transform creates a superposition of states each of which has the bits for some number close to a multiple of $(2^{2L}/s)$. Unless the period is an exact power of two, not all of the superimposed values will be exactly the same, but each will be reasonably close to a multiple of p. It does not particularly matter which state is measured. It is safe to do the measurement and end the computation.

The next step is to interpret the measured value as an exact multiple of $2^{2L}/s$. That step yields the value s, the period of r, and it can be done easily and quickly on a classical computer using a continued fraction expansion, and a search for a square root of 1 modulo N, and a gcd operation.

Shor's algorithm can be adapted to computing discrete logarithms in both multiplicative groups and elliptic curve groups. In each case, more arithmetic circuits are needed. That is especially true for elliptic curves, as Table 6 shows.

Period finding *per se* does not use a huge number of qubits or operations, but error correction adds quite a lot of overhead in time and qubits. Even if a small factorization on quantum circuits could be demonstrated, one that demonstrated all the required circuits and involved all the qubits, it would not presage scaling to a machine with quantum factoring superiority.

There are several suggestions for ways to reduce the quantum resources (qubits, gates, error correction, etc.) for Shor's algorithm [21], [42], [19]. This is an active area of research where incremental improvements are expected over the next decade.

5.3 By the Qubit, Quantum Factoring and Discrete Logarithms

In section 4.9 we examined the state of the art in attacking factoring and discrete logarithms. In this section we show some estimates of the time it would take to run the same kind of attacks with quantum circuits. The estimates show that even the best algorithms have a very slim (if at all) margin with respect to large-scale quantum circuits of the future. The exception is SHA2 hashing, which remains secure even against quantum attacks.

For quantum time estimates, we need to know the required number of qubits and gates. Because Toffoli gates are universal for reversible logic circuits, they are a convenient measure of logical complexity. It is common practice to estimate time based solely on logical qubits and Toffoli gates. The number of logical qubits and the time to operate on a logical qubit must both be scaled by the complexity of the error correction scheme. We use the following estimated parameters for QC execution times:

- Toffoli gate time. Some authors, for algorithm evaluation, arbitrarily assume that a Toffoli gate executes in 1 μsecond. In an error-corrected system, the time will be much longer. Our assumed "real" time for a reliable Toffoli gate is an optimistic 12 μseconds, which is loosely derived from [19].

- Error correction multiplier to convert logical qubits to physical qubits, 3000. This is also based on optimistic design in [19].
- Parallelism. Our estimates assume that arithmetic can be done on short qubit registers in parallel. This is somewhat liked byte parallelism in classical computing. There have been no demonstrations that we know of for register parallelism in quantum circuits, but there is no theoretical objection to it, and it would greatly reduce resources and execution time.

All of these correction factors are wild guesses. We have tried to use the most optimistic numbers from published work, but we acknowledge that without seeing a fully reliable qubit in operation, we have little idea of what we'll see in five years.

Earlier we presented the classical and estimated quantum factoring times in Table 2. That clearly shows the advantage that Shor's algorithm has over classical computing, which would need millions of processor cores to factor even a smallish number of 1000 bits in any reasonable time.

Quantum time to break DSA signatures using discrete log

P outer prime bits	Q inner prime bits	exponent logical qubits	total logical qubits	physical Mqubits	physical break time hours
1024	160	330	2582	9.2	.3
2048	224	458	4964	20	.8
2048	256	522	5028	23	.9
3072	256	522	7280	29	1.3

Table 5: Time to break, quantum, using modified Shor's algorithm. Source: [19] Table 5, page 18; exponent logical qubits is $10 + 2 * InnerPrimeBits$. Total logical qubits includes space for two residues mod P, plus 10% for Addition runways. Line 3, (2048, 256), is interpolated.

The estimates of the number of qubits needed for Shor's algorithm assume that the exponent and modular powers will be represented all at once

Quantum times for solving the
NIST ECDLP curves

bits	logical qubits	Toffoli	physical qubits	days
224	2042	$8.43 * 10^{10}$	$6 * 10^6$	12
256	2330	$1.26 * 10^{11}$	$7 * 10^6$	18
384	3484	$4.52 * 10^{11}$	$10 * 10^6$	60
521	4719	$1.14 * 10^{12}$	$14 * 10^6$	160

Table 6: Quantum computing, time to complete on processor with 12 μsec logical bit gate speed. See [42] table 2.

in a few qubit registers of length N, where N is the number of bits in the target number. There have been proposals to divide-and-conquer that representation by using more, but shorter, quantum registers, each with half the number of bits. Each would run the arithmetic circuits as a "high half" and "low half" register computation simultaneously, and then combine their results to get an estimate of the full period of the group. The parallelism of operating on two qubit registers simultaneously may be difficult to achieve topologically, but it could greatly diminish execution time.

The commonly used public key methods that are not RSA derive their security from the discrete logarithm problem. Although the arithmetic effort for discrete logarithms over multiplicative groups is about the same as factoring in both the classical and quantum domains, more qubits are needed for discrete logarithms than for factoring. More qubits mean more error correction and more delay per quantum circuit. Nonetheless, the time for quantum breaks in these systems is distressingly small if one is interested in security.

In classical computing, elliptic curve groups and "hidden subgroups" (as in DSA) pose a more difficult discrete logarithm problem than do multiplicative groups. These systems use a smaller number of bits to represent their number, so it would seem obvious that they would be easily broken by quantum circuits. That is true for hidden subgroups, but not for elliptic curve groups. Elliptic curve arithmetic requires many more gates for its arithmetic, which includes modular inversion. An additional complication is that in contrast to modular exponentiation, part of which can be efficiently precalculated by a classical computer, almost all the work has to be carried out in the quantum domain. That requires implementing a lot of that arithmetic, and that greatly increases the number of Toffoli gates and adds time to the circuits. Table 5 shows the estimated quantum resources (qubits and Toffoli gates) to break elliptic curve DSA signatures. It is based on a detailed algorithm analysis [42]. From this table we see that elliptic curve discrete logarithms would take small numbers of years to break with quantum circuits. If there were such a machine, and if there were a Moore's Law for quantum computing, then elliptic curve public key systems would be relatively safe for several "doubling times" for the technology.

6 When Will We See a Quantum Attack on Internet Security?

There are some educated and optimistic opinions about near-term quantum factoring superiority [35] and [36], but thus far quantum device demonstrations have not shown anything close to the capabilities that would cause immediate worry among cryptographic practitioners.

The only cryptographic algorithms of interest that are attackable by quantum computing are RSA, DSA signatures, and DH key exchange. These all depend on factoring and discrete logarithms, and the only algorithm that can clearly attack them is Shor's algorithm.

All demonstrations of quantum factoring have used simplifications of the problem that precompute the computationally difficult intermediate results. No full implementation of Shor for any number of bits has been shown. IBM ran superconducting qubit factorizations of the numbers 15, 21, and 35 [7] (only 15 and 21 were completed). Their work demonstrated that the most important gates worked as intended and the answer could be extracted. As a piece of mathematics, it is obviously trivial, but it is a first step on a long road.

Certainly no meaningful demonstration of "cryptographically relevant factoring" can occur before fully reliable qubits are demonstrated.

There have been interesting ideas for reducing the number of qubits and the number of gates in Shor's algorithm. Recent discussion of design improvements for quantum factoring [7] show that there are many ways to match the algorithm to the resources (as we see with classical computing today).

The lack of quantum computing devices has not stopped investigations into efficient implementations of Shor's algorithm on hypothetical hardware. A recent paper posits that the number of physical qubits and the gate fidelity can be greatly improved from the estimates of several years ago [19]. In that scenario, factoring a number of 2048 bits would use 20 million qubits and run for 8 hours. If their ideas work, and if other similarly large reductions in quantum resources and can be realized, quantum factoring might become an attainable goal.

6.1 Moore's Law for Quantum?

Today's computers followed an arc of exponential progress for over 60 years, and this has led to the expectation that quantum computing will somehow do the same thing. Our cloudy crystal ball indicates that even if this happens, it will be a slower exponential than classical computing has seen.

Charles Babbage built a small mechanical computer in the 1820s, and Ada Lovelace developed the ideas for translating an algorithm to a set of computer operations in the 1840s. Their ideas were far ahead of the engineering capabilities of their time. Physicists became aware of the strange world of quantum theory in the 1920s, and Peter Shor's factoring algorithm in 1994 seems to have inspired the field of quantum device engineering. But Shor's algorithm may be as far ahead of the necessary machinery as Lovelace's were.

The computing technology that we use today can be said to have started with the first demonstrated semiconductor transistor in 1947. Computers based on vacuum tubes preceded this, but that technology proved unreliable and unscalable. In contrast, it was only 8 years from the transistor demonstration to the construction of stored program computers. Research on QC devices started almost 30 years ago and has yet to produce even a single reliable qubit.

For an analogy with transistor technology, we might look at the time between the first patent for a transistor (1925), Shockley's initial demonstration of a semiconductor transistor (1947), and the invention of the marvelous MOSFET in 1958 [29]. Twenty years from initial elucidation of the concept to realization of a practical device, eleven years to the commercialization of products based on it, and eleven years to the realization of the optimal device. With that foundation of finance and device technology, the 18 month curve took shape. For quantum, we have about 80 years from the scientific discovery and understanding of quantum phenomena to the idea of building computing devices, twenty years from concept to initial demonstration, and no visible pathway to commercialization. There is no consensus on the technology path going forward. Everything, from basic physical unit to topology and gate construction and an error correction method, is on the table.

A Moore's Law curve has to be supported by scientific discovery, and that does not happen *de novo*. In 1947 there was already a generation of trained electrical engineers; two more would be needed to bring transistor based computers to full fruition; today the number of quantum engineers probably numbers in the low hundreds. The education, discovery, and development pipeline requires financial investment and government incentives at every stage. Google and IBM and a few universities and small companies constitute the entirety of quantum expertise today. It is not a quantum army.

Google aims for a million qubits by 2031 [31], and IBM is aiming for a reliable qubit in 2023 [37]. Lab demonstrations of error detection that increases exponentially with each layer [14] illustrate the possible practicality of their plans, yet both goals seem overly ambitious. But even if they are met, further progress will depend on overcoming a further host of device engineering challenges posed by those skittish and shy qubits.

Even without perfection, the possible applications of quantum computing are myriad. An open question is when will there be some problem for which a quantum computer has superior performance compared to the best classical computer. The question has some importance in predicting when investment in QC will show positive returns. To date, only one algorithm has approached "superiority", and it is an algorithm specifically chosen to favor a peculiar facet of quantum phenomena, namely randomness.

Let us suppose that the quantum technology doubling time is about 4 years. That means we assume it will be 4 years from the demonstration of one logical qubit to the demonstration of performing arbitrary unitary transforms on 2 logical bits. At that rate, it will be 50 years before there will be a quantum computer capable of factoring a 2048 bit number.

All of this means that quantum devices are still very much a work in progress, a work that has yet to define a clear path to success. As the various quantum engineering options are investigated, the research picture is that of "breadth first" search for the best devices. Progress cannot reach the goal of establishing a "Moore's Law" rate of success until engineers converge on devices that have a narrowing yet deepening set of improvements.

Moore's Law for transistors predicts that things will get twice as good every 18 months. Transistors got smaller, they were faster, they used less power, etc. All these attributes are related, so they all improved at about the same rate. We might expect that quantum computers will improve at some fixed rate, eventually, but what might the doubling time be? The 18 month doubling time for semiconductors is a feature of the technologies that go into fabricating transistors (crystal preparation, lithography, etc.), so we might ask what would determine quantum computer doubling time? This is a complicated topic, but it seems safe to say that the doubling time would be much longer than 18 months.

7 Future Directions Towards Quantum Factoring Superiority

Quantum computing may have some near term applications in scientific research that will attract investment and fund development, but quantum factoring is a distant dream. Realization of it will require advances in every aspect of the technology. There are several ideas being explored today, most of them only on paper, but there are good reasons to believe that the coming years will see steady, if only incremental, progress.

- Physical technology. It took many years to find the best material for semiconductors, and we may see that qubits can be made from quite novel particles and configurations. The world of quantum phenomena is strange and difficult to explore; there may be magical self-correcting qubits lurking in some subatomic particle.
- Cooling. Qubits survive only in a very cold (milli-Kelvin) environment, and the control signals bring in disruptive heat. If an algorithm has to run for hours or days, the cooling becomes a big problem. Some proposals would have some of the control run in the cryostat to minimize its heat production.
- Coherence times. The environment is always trying to assault qubits and make them change state. Trapped ions are particularly good at resisting. If there were a way to combine their properties with faster transforms, it would be a major improvement.
- Gate fidelity. Producing a reliable gate is something of a black art today. Shor's algorithm will need extremely reliable gates for the massive amount of arithmetic required for modular reduction. IBM has plans to produce 1 error corrected qubit in 2023 using superconducting bits in a "heavy hex" topology [37]. This configuration is designed to use fixed frequency lasers and to minimize errors due to unintended interference from adjacent energy pulses.
- Topology. Different device technologies might support advantageous layouts of qubits so that the routing of coupling connectors is mini-

mized. That could improve gate fidelity and reduce the overhead of error correction.

- Coupling. The density and layout of coupling connections will affect the running time of algorithms. If qubits can, for example, be laid out in spheres, then higher density coupling might be possible. That would minimize the number of qubit swaps needed for algorithms like multiplication.

- Error correction. Mathematically, we know that if the qubits and gates are reliable enough, error correction is possible, but that does not mean that it is easy or efficient. It might be possible to give up some amount of accuracy for particular circuits and yet still use more reliable circuits to extract the correct answer. The error correction also interacts with the layout and gate design.

  There is significant interaction between error correction and the external control circuitry. This requires a lot of interconnecting wires in a small space. The control circuitry has to execute a fair amount of code that computes the changes needed to bring the system back into a correct state, and this has to be done for each level of error correction. This takes about 10 μsec, but might be brought down to as low as 200 nsec [17]. This number is the most important part of the running time for a qubit intensive algorithm like Shor.

- Mathematical improvements. The brilliance of Shor's algorithm does not mean that it cannot be improved, particularly in the modular exponentiation phase. Straightforward translation of classical circuits to quantum circuits might not be optimal. Mathematical advances might point to improvements.

  Dividing up the arithmetic by using shorter qubit registers in parallel is a generic technique that might find further traction.

  Another interesting proposal is to modify Shor's algorithm for factoring to use fewer bits in the phase estimation by estimating $p+q$ instead of $\phi(N)$.

- Fast, reliable multi-qubit gates. It is difficult to realize a reliable gate, and multi-qubit gates compound the design problem. If it were possible to make such gates for custom arithmetic, such as byte-level arithmetic, Shor's algorithm might need far fewer gate resources.

8 Conclusions

As a new computing paradigm, quantum circuits are fascinating at every level. As more people who revel in tackling hard problems choose to spend their time on quantum computing, more ideas will emerge. At the current time, a large-scale quantum device that could achieve "quantum factoring superiority" and undermine Internet security seems far in the future. Given the difficulty of constructing even very simple quantum circuits, it seems that progress will always be slower than the semiconductor industry saw during the past 70 years, with the consequence that powerful quantum computers are likely 50 or more years in the future.

While cryptographers prepare algorithms for a "post-quantum" world, and while Internet engineers ponder the challenges involved in developing a framework for changing to new algorithms, we should keep in mind that there is no imminent cryptographic catastrophe on the horizon.

9 Thanks

The author gratefully acknowledges ICANN's financial support for this work. Technical help on quantum physics from Joel Howard was invaluable; any errors in explication herein are due to the author's clumsiness. Richard Schroeppel contributed the calculations for the tables of time and effort for cryptographic attacks.

References

[1] FIPS pub 186-4, digital signature standard (DSS), July 2013. National Institute of Standards and Technology, Information Technology Laboratory.

[2] Post-quantum cryptography PQC. *NIST Information Technology Laboratory, Computer Security Resource Center*, 2017. `https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/Call-for-Proposals`.

[3] DST root CA X3 expiration. *LetEncrypt website*, September 2021. `https://letsencrypt.org/docs/dst-root-ca-x3-expiration-september-2021/`.

[4] Japan's Fugaku keeps position as world's fastest supercomputer. *NikkeiAsia*, June 2021. https://asia.nikkei.com/Business/Technology/Japan-s-Fugaku-keeps-position-as-world-s-fastest-supercomputer.

[5] Qisket, open-source quantum development. *github*, 2021. https://qiskit.org/.

[6] National Security Agency. Frequently asked questions: Quantum computing and post-quantum cryptography. *https://media.defense.gov/2021/Aug/04/2002821837/-1/-1/1/Quantum_FAQs_20210804.PDF*, August 2021.

[7] Mirko Amico, Zain H. Saleem, and Muir Kumph. Experimental study of Shor's factoring algorithm using the IBM Q experience. *Phys. Rev. A*, 100, Jul 2019. https://link.aps.org/doi/10.1103/PhysRevA.100.012305.

[8] Philip Ball. Major quantum computing strategy suffers serious setbacks. *Quanta Magazine*, September 2021. https://www.quantamagazine.org/major-quantum-computing-strategy-suffers-serious-setbacks-20210929/.

[9] Joseph C. Bardin, Daniel H. Slichter, and David J. Reilly. Microwaves in quantum computing. *IEEE Journal of Microwaves*, 1(1):403–427, Jan 2021.

[10] Daniel J. Bernstein, Nadia Heninger, Paul Lou, and Luke Valenta. Post-quantum RSA. *International Workshop on Post-Quantum Cryptography*, pages 311–329, 2017.

[11] Thomas Bilitewski, Luigi De Marco, Jun-Ru Li, Kyle Matsuda, William G. Tobias, Giacomo Valtolina, Jun Ye, and Ana Maria Rey. Dynamical generation of spin squeezing in ultracold dipolar molecules. *Phys. Rev. Lett. 126*, March 2021.

[12] Fabrice Boudot, Pierrick Gaudry, Aurore Guillevic, Nadia Heninger, Emmanuel Thome, and Paul Zimmermann. Comparing the difficulty of factorization and discrete logarithm: a 240-digit experiment. *IACR eprint*, 2020. https://eprint.iacr.org/2020/697.pdf.

[13] Gilles Brassard, Peter Hyer, and Alain Tapp. Quantum cryptanalysis of hash and claw-free functions. *Lecture Notes in Computer Science*, pages 163–169, 1998.

[14] Adrian Cho. Physicists move closer to defeating errors in quantum computation. Science magazine, July 2021. `https://www.science.org/content/article/physicists-move-closer-defeating-errors-quantum-computation`.

[15] D. Coppersmith. An approximate Fourier transform useful in quantum factoring. *IBM Technical Report RC19642*, 1994.

[16] Simon J. Devitt, Austin G. Fowler, and Lloyd C.L. Hollenberg. Investigating the practical implementation of Shor's algorithm. *Proceedings of SPIE - The International Society for Optical Engineering*, February 2005.

[17] Austin G. Fowler, Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland. Surface codes: Towards practical large-scale quantum computation. *Physical Review A*, 86(3), Sep 2012.

[18] B. Foxen, C. Neill, A. Dunsworth, P. Roushan, B. Chiaro, A. Megrant, J. Kelly, Zijun Chen, K. Satzinger, R. Barends, F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, S. Boixo, D. Buell, B. Burkett, Yu Chen, R. Collins, E. Farhi, A. Fowler, C. Gidney, M. Giustina, R. Graff, M. Harrigan, T. Huang, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, P. Klimov, A. Korotkov, F. Kostritsa, D. Landhuis, E. Lucero, J. McClean, M. McEwen, X. Mi, M. Mohseni, J. Y. Mutus, O. Naaman, M. Neeley, M. Niu, A. Petukhov, C. Quintana, N. Rubin, D. Sank, V. Smelyanskiy, A. Vainsencher, T. C. White, Z. Yao, P. Yeh, A. Zalcman, H. Neven, and J. M. Martinis. Demonstrating a continuous set of two-qubit gates for near-term quantum algorithms. *Phys. Rev. Lett.*, 125:120–504, Sep 2020. `https://link.aps.org/doi/10.1103/PhysRevLett.125.120504`.

[19] Craig Gidney and Martin Ekerå. How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. *Quantum*, 5:433, April 2021. `https://doi.org/10.22331/q-2021-04-15-433`.

[20] Markus Grassl, Brandon Langenberg, Martin Roetteler, and Rainer Steinwandt. Applying Grover's algorithm to AES: quantum resource estimates. *PQCrypto*, pages 29–43, February 2016.

[21] Thomas Haner, Martin Roetteler, and Krysta M. Svore. Factoring using 2n+2 qubits with toffoli based modular multiplication, 2017. `https://arxiv.org/abs/1611.07995`.

[22] Werner Heisenberg. *Physics and Beyond*. Harper & Row, 1971.

[23] Jonathan Hui. QC - how to build a quantum computer with superconducting circuit? *Medium.com*, January 2019. `https://jonathan-hui.medium.com/qc-how-to-build-a-quantum-computer-with-superconducting-circuit-4c30b1b296cd`.

[24] Geoff Huston and Joo Damas. DNSSEC with RSA-4096 keys. The ISP Column: A column on things Internet, October 2021. `https://www.potaroo.net/ispcol/2021-10/rsa.html`.

[25] Charlie Kaufman. Internet Key Exchange (IKEv2) Protocol. RFC 4306, December 2005.

[26] R. Landauer. Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development*, 5(3):183–191, 1961.

[27] Carlile Lavor, Leon R. U. Manssur, and R. Portugal. Shor's algorithm for factoring large integers. *arXiv: Quantum Physics*, 2003.

[28] A.K. Lenstra, H.W. Lenstra, M.S. Manasse, and J.M. Pollard. The development of the number field sieve. *Lecture Notes in Mathematics, vol 1554*, 1993.

[29] Bo Lojek. History of semiconductor engineering, 2007. `https://archive.org/details/historysemicondu00loje_697`.

[30] Junling Long, Tongyu Zhao, Mustafa Bal, Ruichen Zhao, George S. Barron, Hsiang sheng Ku, Joel A. Howard, Xian Wu, Corey Rae H. McRae, Xiu-Hao Deng, Guilhem J. Ribeill, Meenakshi Singh, Thomas A. Ohki, Edwin Barnes, Sophia E. Economou, and David P. Pappas. A universal quantum gate set for transmon qubits with strong ZZ interactions, 2021.

[31] Erik Lucero. Unveiling our new quantum AI campus. 2021. `https://blog.google/technology/ai/unveiling-our-new-quantum-ai-campus/`.

[32] John Preuss Mattsson, Ben Smeets, and Erik Thormarker. Quantum-resistant cryptography, 2021. `https://arxiv.org/abs/2112.00399`.

[33] John Preuss Mattsson, Ben Smeets, and Erik Thormarker. Quantum technology and its impact on security in mobile networks. Ericsson Technology Review, December

2021. https://www.ericsson.com/4ae3c7/assets/local/reports-papers/ericsson-technology-review/docs/2021/ensuring-security-in-mobile-networks-post-quantum.pdf.

[34] Timothy Prickett Morgan. A first peek at chinas sunway exascale supercomputer. *TheNextPlatform*, February 2021. url=https://www.nextplatform.com/2021/02/10/a-sneak-peek-at-chinas-sunway-exascale-supercomputer/.

[35] Michele Mosca. Cybersecurity in an era with quantum computers: will we be ready? *IACR Eprint*, 2015. https://eprint.iacr.org/2015/1075.pdf.

[36] Michele Mosca and Nick Sullivan. Michele Mosca and Nick Sullivan in conversation. Cloudflare, video, September 2021. https://cloudflare.tv/event/4eORUHIua5EUSukb9pRLvZ.

[37] Paul Nation, Hanhee Paik, Andrew Cross, and Zaira Nazario. The ibm quantum heavy hex lattice. *IBM Tech Blog*, 2021. https://www.research.ibm.com/blog/heavy-hex-lattice.

[38] Medicine National Academies of Sciences Engineering˙ *Quantum Computing: Progress and Prospects*. The National Academies Press, Washington, DC, 2019.

[39] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[40] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, August 2018.

[41] Eric Rescorla and Allan M. Schiffman. The Secure HyperText Transfer Protocol. RFC 2660, August 1999.

[42] Martin Roetteler, Michael Naehrig, Krysta M. Svore, and Kristin Lauter. Quantum resource estimates for computing elliptic curve discrete logarithms. Cryptology ePrint Archive, Report 2017/598, 2017. https://ia.cr/2017/598.

[43] Scott Rose, Matt Larson, Dan Massey, Rob Austein, and Roy Arends. Protocol Modifications for the DNS Security Extensions. RFC 4035, March 2005.

[44] P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[45] Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, and Yarik Markov. The first collision for full SHA-1. Cryptology ePrint Archive, Report 2017/190, 2017. `https://ia.cr/2017/190`.