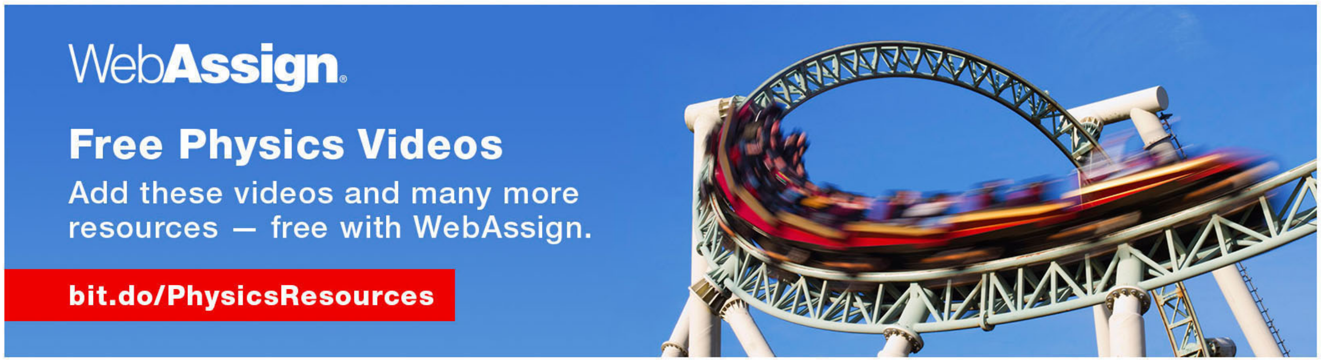
Shor's factoring algorithm and modern cryptography. An illustration of the capabilities inherent in quantum computers

Edward Gerjuoy

Citation: American Journal of Physics **73**, 521 (2005); doi: 10.1119/1.1891170
View online: http://dx.doi.org/10.1119/1.1891170
View Table of Contents: http://scitation.aip.org/content/aapt/journal/ajp/73/6?ver=pdfcov
Published by the American Association of Physics Teachers


WebAssign.
Free Physics Videos
Add these videos and many more resources — free with WebAssign.
bit.do/PhysicsResources

Shor's factoring algorithm and modern cryptography. An illustration of the capabilities inherent in quantum computers

Edward Gerjuoy
Department of Physics, University of Pittsburgh, Pittsburgh, Pennsylvania 15260

(Received 11 May 2004; accepted 11 February 2005)

The security of messages encoded via the widely used RSA public key encryption system rests on the enormous computational effort required to find the prime factors of a large number N using classical (conventional) computers. In 1994 Peter Shor showed that for sufficiently large N, a quantum computer could perform the factoring with much less computational effort. This paper endeavors to explain, in a fashion comprehensible to the nonexpert, the RSA encryption protocol; the various quantum computer manipulations constituting the Shor algorithm; how the Shor algorithm performs the factoring; and the precise sense in which a quantum computer employing Shor's algorithm can be said to accomplish the factoring of very large numbers with less computational effort than a classical computer. It is made apparent that factoring N generally requires many successive runs of the algorithm. Our analysis reveals that the probability of achieving a successful factorization on a single run is about twice as large as commonly quoted in the literature. © *2005 American Association of Physics Teachers.*

[DOI: 10.1119/1.1891170]

I. INTRODUCTION

Recently published papers[1,2] have attempted to explain how a quantum computer differs from a classical, that is, a conventional, computer. However, neither of these papers offers a detailed discussion of the factoring algorithm developed by Peter Shor[3] in 1994, although this algorithm, whose implementation could frustrate one of the most widely used modern methods of encrypting messages, provides the most impressive known illustration of the increased computing power potentially attainable with quantum computers. This paper furnishes a self-contained explanation of Shor's algorithm, as well as of the algorithm's relevance to modern cryptography. Shor's original paper[3] and other publications discuss the Shor algorithm in a manner suitable for quantum computing specialists.[4–6] Less technical presentations more suitable for the nonspecialist readers of this journal also are available.[7–9] Various Web sites link to a wide variety of publications in the quantum computing literature, organized under numerous suitable headings including Shor's algorithm.[10]

The contents of this paper can be summarized as follows. Section II first describes the basic elements of classical cryptography, where *keys* are employed to encipher and/or decipher messages to prevent those messages from being read by anyone other than their intended audience. Section II then explains in detail the enciphering and deciphering procedures in the RSA system,[11–13] an important modern scheme for sending secret messages. These explanations and illustrations require the presentation of the number theory underlying the RSA system. Without some understanding of this underlying number theory, the ability of the RSA system to transmit secure messages seems magical. Section II goes on to explain how the security of the RSA system critically depends on the fact that using classical computers to factor large numbers requires huge outlays of computer resources and time. The relevance of Shor's factoring algorithm to the security of the RSA system thereby becomes evident. After briefly summarizing the relevant properties of quantum computers, Sec. III fully describes and illustrates Shor's algorithm. Section III also explains the precise sense in which a quantum computer employing Shor's algorithm can be said to accomplish the factoring of very large numbers with less computational effort than a classical computer. Many details of the underlying number theory, which are important not only for understanding why the RSA system works, but also for comprehending how Shor's algorithm enables the factorization of large numbers, are discussed in the appendices.

Except for its discussion of the Shor algorithm, which is designed for a quantum computer, this paper is concerned solely with classical computers. In particular, I assume that the enciphering and deciphering procedures described in Sec. II are performed with classical computers, which experience has demonstrated is practical with numbers of the size presently being used as RSA keys. The possible use of quantum computers to perform this enciphering and deciphering and other aspects of quantum cryptography and/or computation aside from Shor's algorithm are beyond the scope of this paper. Also beyond the scope of this paper are the observed and anticipated difficulties involved in actually constructing functioning quantum computers. These difficulties are well discussed in the literature, for example, in Ref. 6.

II. CRYPTOGRAPHY, KEY DISTRIBUTION AND NUMBER THEORY

As Ekert has observed,[14] "Human desire to communicate secretly is at least as old as writing itself and goes back to the beginnings of our civilization." The full history of secret communication until about 1965 is recounted by Kahn.[15] Developments after about 1965, including those advances in secret communication to which Shor's algorithm pertains, are described by Singh,[16] who also (but less fully than Kahn) reviews the pre-1965 history. Kahn[15] carefully defines the terms *plaintext* (the original uncoded message), *cryptogram* (a writing in code, for example, the enciphered message), *key* (the information or system employed to encipher the plaintext message), *cryptography* (the acts of enciphering the plaintext into a cryptogram, and/or of deciphering the cryptogram by someone who knows the key), and *cryptanalysis* (the art/science of code breaking, that is, of ferreting the key from the cryptogram). This paper adopts Kahn's terminology.

Until about 1975 cryptographers employed *symmetric* or *private* (also known as *secret*) key systems only.[17] In such systems the key used by Alice (conventionally the sender of the cryptogram) to encipher the message is the same as the key Bob (conventionally its receiver) employs to decipher the cryptogram. One of the simplest type of symmetric keys (which also appears to have been the earliest type to be employed, dating back to nearly 2000 B.C.[18]) is termed *substitution*. In substitution-key cryptography the cryptogram is constructed from the plaintext by replacing each letter (of the alphabet) in the plaintext with some other chosen expression. The replacement expression can be another letter of the alphabet, or a symbol of some kind, or an arbitrary combination of letters and symbols. The same letter of the alphabet in different portions of the plaintext may be replaced by different expressions. The key is the chosen replacement scheme. In the very simplest substitution keys, which for the purpose of this paper may be termed *unique substitution*, Alice and Bob agree that the replacements will be unique and one to one, that is, any given letter in the plaintext always is replaced by the same expression, and different letters of the alphabet are replaced by different expressions. Although cryptograms constructed via unique substitution keys can seem impregnable, especially when the key involves unusual or unfamiliar symbols, they actually are readily deciphered taking advantage of the peculiarities of the language (assumed to be known or correctly guessed) in which the plaintext had been written, as 15th century Arab cryptologists already knew.[19] The writings of Edgar Allan Poe[20] and Arthur Conan Doyle[21] provide celebrated fictional cryptanalyses of unique substitution cryptograms (in these cases with English plaintexts) where letters of the alphabet had been replaced by symbols. In the most transparent unique substitution cryptograms a single alphabet letter is substituted for each plaintext letter, consistent with a preselected *cipher alphabet*. Cryptograms constructed in this fashion, but employing a different cipher alphabet each day, are regularly published in many newspapers as puzzles to be deciphered by readers using, for example, the fact that in English the letter e is by far the most frequent.[22]

But substitution cryptograms need not be constructed with unique keys. Moreover, nonunique substitution cryptograms can be and have been made very difficult to cryptanalyse. A famous illustration of this last assertion is provided by the Enigma machine employed by the German army during World War II, which constructed cryptograms where each letter was replaced by a single letter as in newspaper cryptograms, but where the cipher alphabet employed to encipher any given plaintext varied not merely from day to day, but also from one plaintext letter to the next, in accordance with a predetermined randomly selected complicated key.[23] Whether or not very difficult to cryptanalyse, however, all substitution and other symmetric key cryptographic systems have a deficiency known as the *key distribution problem*, as numerous authors (e.g., Ekert[14]) have observed: Before Alice and Bob can begin exchanging cryptograms, they must exchange—in a non-encrypted form—the information necessary to establish their key. They cannot be confident that this information exchange has not been intercepted unless the exchange takes place in the same room, and perhaps not even then.[24] Symmetric key cryptographic systems also have the related deficiency that if Bob wants to receive secret messages from more than one Alice, then he either must set up different keys with each Alice or else risk the possibility that one Alice will intercept and readily decipher a message sent to Bob by another Alice.

These deficiencies are avoided in *asymmetric* (commonly termed *public*) key systems, where the key Alice employs to encipher her message is not the same as the key Bob employs to decipher the cryptogram he receives. The differences between symmetric and asymmetric key systems can be visualized in terms of a safe: With a symmetric system the key Alice employs to open the safe and lock her message inside it is the same as the key Bob employs to open the safe and remove the message. With an asymmetric system Alice's key enables her to open the safe just enough to insert her message, but no more. Only Bob's key can open the safe's door sufficiently to permit message removal.[25] Indeed in the RSA system,[7] one of the most commonly employed public key systems, Alice's enciphering key is made public; it is not at all secret but rather is available to any Alice who wishes to send Bob an encrypted message. Bob's deciphering key remains his secret. The nature of the RSA system is given in the following.

A. The RSA public key system

Bob creates his RSA public key system in the following fashion:[11] He first selects two different large prime numbers p and q, and then computes their product $N=pq$. Next he also computes the product $\phi=(p-1)(q-1)=N+1-(p+q)$, and then selects a positive integer e which is coprime to ϕ (meaning that e and ϕ have no common prime factors other than 1). Finally he computes a positive integer d such that the product $L=de$ has the remainder unity when divided by ϕ. Bob now has all the required components of his public key system except the specified convenient-to-use symmetric key whereby any Alice can convert her plaintext into a cryptogram consisting of a sequence C of positive integers c, which she then will further encrypt (via Bob's proclaimed procedure) into the sequence S of positive integers s actually sent to Bob.

The nonsecret components of Bob's public key system, which Bob now is ready to broadcast for the benefit of one and all, are (*i*) the positive integers N and e, which here will be termed the *key number* and *encryption exponent*, respectively; (*ii*) the details of the symmetric key Alice will use to construct her C, and which Bob also will use to reconstruct Alice's original plaintext message once he has deciphered S and thereby recovered C (the only restriction on C is that every c must be less than N); and (*iii*) the surprisingly simple procedure for constructing the elements s of S from the elements c of C, namely s is the integer remainder when c^e is divided by N.[11] Note that the symmetric key which Alice and Bob share now is completely public; there is no attempt whatsoever to keep it secret. Similarly, Alice transmits her finally enciphered cryptogram S to Bob via perfectly open communication channels, for example, by email. The secret procedure by which Bob extracts the original C from S parallels, but is not the direct inverse of, the public procedure which constructed S from C, namely, for each s Bob computes the integer u which is the remainder when s^d is divided by N.[11] Bob is confident that because he has kept the decryption exponent d secret, he and only he possesses the secret key that enables deciphering of S.

At this juncture it is instructive to illustrate the encryption and decryption of messages in this RSA public key system of Bob's, in particular the dread message to Bob from Alice that "The FBI came." To do so, it first is necessary to specify the symmetric key. A possible easily usable symmetric key, which Singh suggests,[26] requires Alice to replace each letter of the alphabet by its ASCII equivalent. ASCII[27] is the protocol that converts computer keyboard strokes into the seven-bit electrical impulses that transmit our email; each impulse is the binary representation of a positive integer (less than $2^7=128$). In ASCII the 26 upper case letters A through Z are represented by the integers (in base 10) 65 through 90; the corresponding lower case letters are represented by the integers 97 through 122; a space between words is represented by the integer 64; the ASCII integer representations of other communication symbols, for example, the period or the comma, are irrelevant here. Similarly, it is simpler to ignore the distinction between upper and lower cases, and to represent the letters A through Z by the integers 2 through 27 respectively, saving the integer 1 for the space between words.

Thus, Alice's C, corresponding to her dread message, is 21, 9, 6, 1, 7, 3, 10, 1, 4, 2, 14, 6. These integers are written in base 10, and, unless otherwise noted, integers will be written in base 10 in the remainder of this paper. That Alice (with Bob's blessing) may choose to write her integers in another base, or may transmit her S to Bob via email (where the digits 0 through 9 she uses to write her base 10 integers will be converted into electrical impulses that are the ASCII seven-bit binary equivalents of the base 10 integers 48 through 57, respectively[27]) does not affect the validity of any conclusions given in the following.

Next it is necessary to choose the pair p and q of primes whose product yields the N our hypothetical Bob had broadcast for Alice's use. I will choose the pair 5 and 11, which makes $N=55$, a conveniently small number for illustrative purposes, but still large enough to satisfy the requirement that N exceeds every c in C. The quantity $\phi=4\times10=40$. Thus I now can choose $e=23$, consistent with the requirement that e be coprime to ϕ. To complete Bob's public key, we need a positive integer d such that when de is divided by ϕ, the remainder is unity. The integer $d=7$ fits the bill, as the reader can verify. The convenient method which Bob can use to find d in actual RSA practice is presented in Appendix D 4.

Finally, I am in a position to illustrate how Alice constructs her cryptogram S from C. To do so efficiently, however, it is desirable to introduce the modular arithmetic notation employed in number theory.[28]

B. Modular arithmetic formulation of the RSA public key system

If the positive integer b is the remainder when the positive integer a is divided by the positive integer m, then $a-b$ is exactly divisible by m. In number theory, if a difference of two integers a and b (each of which may be positive or negative) is exactly divisible by the positive integer m, then we say a and b are congruent modulo m, and write[28]

$$a\equiv b \pmod{m}. \tag{1}$$

Thus the procedure for obtaining the elements s of S from the elements c of C can be written as

$$c^e\equiv s \pmod{N}. \tag{2}$$

Similarly Bob's secret key procedure for deciphering S can be written as $c=u$, where

$$s^d\equiv u \pmod{N}. \tag{3}$$

Finally, the expression yielding d from e is

$$de\equiv 1 \pmod{\phi}. \tag{4}$$

Because Eq. (1) means there is no remainder when $a-b$ is divided by m, Eq. (1) can be restated as

$$a-b\equiv 0 \pmod{m}. \tag{5}$$

But if $a-b$ is exactly divisible by m, then so is $a-b-m$, that is, if Eq. (1) holds, then it is also true that

$$a\equiv b+m \pmod{m}. \tag{6}$$

Equations (1) and (6) imply that Eq. (2), though perfectly correct, does not uniquely determine s without the additional condition that s is a positive integer less than N, which then guarantees that s is indeed the integer remainder when c^e is divided by N. Similarly Eq. (3) does not uniquely specify u without the additional condition that u is a positive integer less than N.

The use of congruences eases Alice's task of constructing her cryptogram S from C via Eq. (2). In particular, for the key number $N=55$, encryption exponent $e=23$, and the example C, Alice's S is 21, 14, 51, 1, 13, 27, 10, 1, 9, 8, 49, 51. By using the decryption exponent $d=7$ in Eq. (3), Bob can then decrypt this S into Alice's original C. These calculations are illustrated in Appendix A. Appendix C contains the proof that the RSA system enables Bob to correctly decipher every S Alice transmits and the proof of the fact that Eqs. (2)–(4) imply that $u=c$ when $N=pq$ and $\phi=(p-1)(q-1)$.

C. Cryptanalysis of RSA system messages

The example S was obtained from the example C by successively inserting each individual c into Eq. (2). But insertion of the same c into Eq. (2) always yields the same s. For example, because both the third and last numbers in C are 6, both the third and last numbers in S turn out to be 51. In other words, our example S is identical to the S into which Alice's original plaintext would have been enciphered using an appropriate unique substitution key of the sort described at the beginning of Sec. II. Of course, with actual RSA key numbers N the actually encountered s in S typically will be very large, not the two digit numbers less than 55 of our example S. Nevertheless, it is now apparent that, despite the RSA system's number theoretic sophistication, if Alice continues to routinely encipher [via Eq. (2)] her plaintext into individual elements s one letter at a time, then the various S she transmits to Bob could be readily cryptanalysed, without any need to guess Bob's decryption exponent d or to employ Eq. (3) at all. In particular, because Alice does not attempt to keep secret the messages S she sends to Bob, the relative frequencies and other characteristics of the various s in her messages, assumed to be written in English, are readily ascertainable. Consequently the standard deciphering techniques[19–22] applicable to symmetric unique substitution keys will work quite well (for example, the observation that a relatively infrequent s_1 almost invariably is followed by another s_2 suggests s_1 is q and s_2 is u).

However, there is no requirement that Alice obtain S by successively inserting each individual c into Eq. (2). For instance, Alice can simply and efficiently obtain a very much more difficult to cryptanalyse S' by first converting her C into a new C' composed of integers c' constructed from large blocks of the original c entries; the only limit on the size of these blocks is that each c' must be less than the key number N. Obviously, there can be essentially no available useful information, for the purposes of cryptanalysis, about the relative frequencies with which blocks of say 40 letters occur in English, especially when during the enciphering those letters can be interrupted by punctuation marks, as well as by superfluous digit combinations. Thus, returning to our illustrative C, whenever Bob's key number N exceeds 10^{40}, Alice could make her illustrative S very much more difficult to decipher by inserting into Eq. (2) not the individual 12 c's comprising the illustrative C, but rather the single 40-digit integer $c' = 2143790906449201290703821001045802143806$. This c' is composed of 20 pairs of digits where, reading from left to right, the pairs whose magnitudes are less than 28 comprise the original 12 c's in their correct order in C, but each pair exceeding 28 is superfluous and randomly chosen. Alice's insertion of this c' into Eq. (2) would yield a single s' comprising Alice's entire new sequence S' to be transmitted to Bob. This s' would have no discernible features related to the presence of those superfluities; yet Bob, after recapturing c' from this s' using Eq. (3), would instantly be able to recognize and discard the superfluous pairs, that is, he would have no difficulty reconstructing Alice's original ''The FBI came'' from his recaptured c'. Alice could similarly break up, into successively transmitted blocks of 40 digits, messages longer than our illustrative ''The FBI came.''

This 40-block scheme is by no means the only conceivable means of replacing a letter by letter S by an S' whose elements s' bear no useful relation to the characteristics of the language in which the original plaintext message was written. Moreover, modern RSA key numbers N permit blocks considerably larger than merely 40 decimal digits. In short, available and feasible encryption systems make the likelihood that Alice's RSA-system messages could be deciphered via the aforementioned techniques, even after receipt by the would-be decipherer of many openly transmitted messages of hers, virtually zero. On the other hand, techniques that depend primarily on language properties are not the only conceivable means whereby a third party might seek to decipher Alice's RSA messages to Bob. Descriptions of alternative deciphering schemes are beyond the scope of this paper. It is sufficient to state that no known means of deciphering RSA messages is computationally more practical than decipherment via factorization of the key number N. In particular, in 1996 a very thorough examination[29] of the security of the RSA system found that ''While it is widely believed that breaking the RSA encryption scheme is as difficult as factoring the key number N, no such equivalence has been proven.'' I am not aware of any later publications that contradict this conclusion. The relevance of being able to factor N is that once the primes p and q factoring N are known, the value of ϕ immediately is yielded by the relation $\phi=(p-1)(q-1)$. Knowing ϕ and Bob's publicly broadcast encryption exponent e, we can easily determine Bob's originally secret decryption exponent d (see Appendix D 4).

D. Modern RSA systems and factoring $N=pq$ with classical computers

It follows that Bob can be confident in the security of his RSA system, provided there is no practical likelihood that a would-be codebreaker will be able to deduce the prime factors p and q of N from the publicly known quantities N and e. The basis for Bob's confidence is the difficulty of factoring, with classical computers, numbers N of the astonishingly large magnitudes typically employed in modern RSA keys. According to recent publications by RSA Security, the company founded by the inventors of the RSA system, key numbers N of 1024 binary digits are now the recommended and popularly employed sizes for corporate use;[30] 1024 binary digits correspond to 309 decimal digits.

The most obvious way to factor a large integer N that is not a prime is to perform the sequence of divisions of N by the integers $2,3,\ldots \leqslant \sqrt{N}$ until a factor of N is found. If N has only two prime factors, each of the order of $\sqrt{N}$, then approximately $\sqrt{N}$ divisions would be required to find the factors of N. Thus, as Ekert[31] has pointed out, this straightforward procedure cannot possibly be relied on to yield the prime factors of really large numbers N, numbers of 100 decimal digits say, which, although very large by ordinary standards, are very much smaller than the present RSA-recommended key numbers of 309 decimal digits. For a 100 decimal digit number N, that is, for N of the order of 10^{100}, approximately 10^{50} divisions may be required to ensure factoring by this straightforward procedure. Even if the average time for a single division is as small as 10^{-12} s, already a very small time for today's fastest computers,[32] the total time required to factor $N \sim 10^{100}$ in this fashion would be $\sim 10^{38}$ s, a duration much longer than 4.32×10^{17} s $=13.7$ billion years, the present estimate of the age of the universe.[33]

Nevertheless, in 1994 the 129 decimal digit public key number N known as RSA-129 was factored after only eight months of number crunching, thereby winning a symbolic \$100 prize Martin Gardner had offered in 1977, shortly after RSA-129 was first made public.[34] This accomplishment was made possible by the ingenuity of mathematicians who were able to devise factoring procedures far more powerful than the brute force procedure we just described. In particular, RSA-129 was factored using the *quadratic sieve*.[35] In 1999 RSA-155 (corresponding to 512 binary digits) was factored after no more than seven months of computing time,[36] using the even more powerful *number field sieve*.[37,38] This factorization of RSA-155, in response to the *Factoring Challenge*[36] started in 1991 by RSA Security, is the primary reason that RSA Security increased its recommended key number N size from the previous RSA-155 to the present RSA-309.[30] RSA Security now recommends a key number size of 2048 binary digits (RSA-617) for extremely valuable keys.[30] The wisdom of this recommendation is manifested by two recent successful factorizations in response to the Factoring Challenge. Factorization of RSA-160 (corresponding to 530 binary digits) was announced[39] on 1 April, 2003. The announcement stated that RSA-160 was factored in less time than RSA-155, and made use of fewer computers in parallel. The announcement that RSA-174 (corresponding to a number of 576 binary digits) had been factored came on 3 December, 2003, only eight months later.[40] As of this writing the time and computer facilities needed to factor RSA-174 have not been released.

That RSA-155 was factored with the expenditure of about the same amount of computing time as RSA-129 reflects not only the improved power of the number field sieve over the quadratic sieve, but also the fact that classical computers had greatly improved in speed during the mere five year interval from 1994 to 1999. This improvement is expected to continue, as the comparison of the factorization times of RSA-155 and RSA-160 exemplifies. Thus it is estimated that by 2009 a computer costing no more than \$10 million will be able to factor RSA-309 in less than one month.[30] Correspondingly, it is anticipated that by 2010 the standard RSA key number size will be 2048 binary digits,[30] and by 2030 will be 3072 binary digits[30] (corresponding to RSA-925). Inherent in these anticipations is the well-founded belief that if classical computing is all that is available, then RSA public key systems can be kept secure via increases of key number size no matter how much classical computers improve, because the magnitude of the computing effort needed to factor a large number N increases very rapidly with increasing N.

An analysis of the number field sieve (presently the most efficient general-purpose factoring technique[38] for numbers N of modern key number size) leads to the conclusion that the number ν of bit operations required to factor a large key number N with a classical computer is expected to increase with N no less rapidly than[38,41]

$$\begin{aligned}\nu(N) &= \exp[(1.90)(\ln N)^{1/3}(\ln\ln N)^{2/3}]\\ &\cong \exp[(1.32)L^{1/3}(\log_2 L)^{2/3}],\end{aligned} \tag{7}$$

where $L=\log_2 N$, and a bit operation[42] denotes an elemental computer operation, e.g., the addition of two bits. The growth of the right side of Eq. (7) as a function of L is termed[43] *subexponential*, that is, more rapidly than any power of L, but less rapidly than $\exp(L)$. For a given computer, that is, for a specified number of processors of specified speeds, the time $\tau(N)$ to perform the factoring should be proportional to $\nu(N)$. Thus Eq. (7) predicts that the aforementioned hypothesized \$10 million computer, which in 2009 will be able to factor RSA-309 in less than a month (say two weeks), still will require about 60 million years to factor RSA-617. We add that even for very large N, the computing effort required to find a pair of primes p and q of magnitude $\sim\sqrt{N}$ is surprisingly small,[44] so that the ability to keep ahead of classical computer factorization abilities via steadily increasing key number N sizes is not limited by any impracticality in finding key numbers $N=pq$. Similarly, although it may be thought that the increasing encryption and decryption times inevitably associated with N ultimately will provide a practical upper bound on its size, as of the foreseeable future any such bound, although it may exist in principle, will be utterly inconsequential.[45]

In summary, Bob's confidence in the present and future security of the RSA systems appears to be justified if classical computing is all that is available. On the other hand, his confidence in the continued security of RSA systems would not be well founded if quantum computers able to employ Shor's algorithm could be constructed, as we now demonstrate.

III. FACTORING USING SHOR'S ALGORITHM

Shor's algorithm, which is designed to take advantage of the inherent potential of quantum (in contrast to classical) computers, exploits a factorization method that differs from the sieves that presently are employed for large key number factorization. We will begin with this factorization method. The relevant properties of quantum computers are summarized in Sec. III B. Section III C then carefully describes and illustrates Shor's algorithm. Some concluding remarks pertinent to the algorithm are given in Sec. IV. Sections III and IV help clarify the claim that factoring increasingly large key numbers ultimately should require less computational effort, that is, ultimately should be more feasible with quantum computers employing Shor's algorithm than with classical computers.

A. Factoring $N=pq$ using the order property of integers coprime to N

Let n denote a positive integer coprime to $N=pq$, where p and q are two distinct large primes. For any such n, let f_j, $j=1,2,3,...$, be the remainder when n^j is divided by N. Then, as with Eqs. (2) and (3), f_j is uniquely specified by

$$n^j \equiv f_j \pmod{pq} \tag{8}$$

together with the condition $0<f_j<N$. As explained in Appendix B, for any n

$$n^{\phi}=n^{(p-1)(q-1)}\equiv 1 \pmod{pq}, \tag{9}$$

implying $f_\phi=1$ for all n. For any given n, however, there may exist other integers $1\leqslant j\leqslant \phi=(p-1)(q-1)$ for which $f_j=1$. The smallest such j, to be denoted by r, is termed[46] the *order* of n modulo pq. Thus, using Eqs. (1) and (5), Eq. (8) for $j=r$ can be restated as

$$n^r-1\equiv 0 \pmod{pq}. \tag{10}$$

Suppose now the order r of some integer $n<N$ and coprime to N is known (how r actually is determined is discussed in the following). Suppose further that r is even, necessary in order that $n^{r/2}$ be an integer and thus meaningfully employable in congruences. Then Eq. (10) implies

$$(n^{r/2}-1)(n^{r/2}+1)\equiv 0 \pmod{pq}. \tag{11}$$

Because by definition r is the smallest power of n for which Eq. (10) holds, the factor $(n^{r/2}-1)$ on the left side of Eq. (11) cannot be exactly divided by pq, that is,

$$n^{r/2}-1\not\equiv 0 \pmod{pq}. \tag{12}$$

The second factor on the left side of Eq. (11) is not subject to any such restriction, that is, it is possible that

$$n^{r/2}+1\equiv 0 \pmod{pq}. \tag{13}$$

It is not necessary that Eq. (13) hold. It is possible that

$$n^{r/2}+1\not\equiv 0 \pmod{pq}. \tag{14}$$

If both Eqs. (12) and (14) hold, we have the case that the product on the left side of Eq. (11) is exactly divisible by pq, although neither factor in this product is exactly divisible by pq. It follows that, to avoid contradiction, one of the factors in the product on the left side of Eq. (11), such as the factor $(n^{r/2}-1)$, must be divisible by p but not by q, while the other factor, $(n^{r/2}+1)$, is divisible by q but not by p. When the order r of n modulo N is even, therefore, and Eqs. (12) and (14) both hold, Bob's proclaimed key number $N=pq$ is immediately factored by computing the following two greatest common divisors (gcds): N with $(n^{r/2}+1)$, and N with $(n^{r/2}-1)$. Alternatively, one can factor N by first computing

q, say, as the gcd of N with $(n^{r/2}+1)$, and then determining p via division of N by this q. The convenient Euclidean algorithm for finding the gcd of two integers is described in Appendix D 1.

The probability that a randomly selected $n<N=pq$ and coprime to N will have an even order r satisfying Eq. (14) is approximately $\frac{1}{2}$.[47] Moreover, as explained in Appendix D 1, calculating the gcd of a pair of large numbers using classical computers is a straightforward procedure requiring negligible computing time compared with the factorization times given in Sec. II D. Therefore, the feasibility of factoring a large $N=pq$ via the procedure described in the preceding paragraph depends primarily on the feasibility of determining the order r of n modulo N for arbitrarily selected n. With classical computers this determination requires solving the *discrete log problem*.[48] Experience has shown[49] that classical factoring of large $N=pq$ via solutions to the discrete log problem is not more feasible than factoring N using the sieves discussed in Sec. II D.

On the other hand, with quantum computers determining r, and thereby factoring N, becomes feasible using the periodicity property of the sequence f_j, $j=1,2,3,...$, defined via Eq. (8). Namely, it is proved in Appendix A that for any n all the integers $f_1, f_2, \ldots, f_{r-1}, f_r=1$ are different, but that for each j in the range $1 \leqslant j \leqslant r$ and every positive integer k, we have $f_j=f_{j+r}=f_{j+2r}=\cdots=f_{j+kr}=f_{j+(k+1)r}=\cdots$. In other words, the sequence f_j, $j=1,2,3,...$, is periodic with period r. For example, returning now to our illustrative $N=55$ key number, for $n=16$ the order $r=5$ and the sequence f_j is (starting with $j=1$) 16, 36, 26, 31, 1, 16, 36, 26, 31, 1, 16, 36,... . Similarly for $n=12$, the order $r=4$ and the sequence f_j is 12, 34, 23, 1, 12, 34, 23, 1, 12, 34,... . Shor's algorithm employs the quantum computer analog of Fourier transformation to extract the order r from a quantum computer wave function that has been specially constructed to exhibit this r periodicity for some randomly selected n. Moreover, the computational effort required to determine r using Shor's algorithm increases with N no more rapidly than some power of N, and increases much more slowly with N than does the effort required to factor N using a classical computer. Classical computer factoring via the solution of the discrete log problem does not result in a slower increase of $\nu(N)$ with N than Eq. (7), because with such computers the number of bit operations required to calculate a Fourier transform is proportional to $NL=L2^L$, that is, increases with N even more rapidly than does the right side of Eq. (7).[50]

I emphasize that once a suitable r has been determined using Shor's algorithm, the factorization of N using Eqs. (12) and (14) can be routinely performed on a classical computer. Referring to our illustrations in the preceding paragraph, for the choice $n=12$ Shor's algorithm will yield $r=4$. Then from the sequence f_j for $n=12$ we need to insert $f_{r/2}=f_2=34$ (which is congruent to 12^2 modulo 55) into Eq. (11). Because Eqs. (12) and (14) both are satisfied for this f_2, we immediately know that $f_2+1=35$ must be divisible by one of the factors of 55 (in this case 5, as we would determine by computing the gcd of 35 and 55), and that $f_2-1=33$ must be divisible by the other factor of 55 (in this case 11), as we would determine either by computing the gcd of 33 and 55 or (more simply) by direct division of 55 by its already determined factor of 5.

B. Quantum computers

Mermin[1] and Grover[2] explain how quantum computers differ from classical computers. I will briefly summarize the background needed about quantum computers to comprehend the functioning of Shor's algorithm, beginning with a quote from Grover:[2] "Just as classical computing systems are synthesized out of two-state systems called bits, quantum computing systems are synthesized out of two-state systems called *qubits*. The difference is that a bit can be in only one of the two states at a time. On the other hand a qubit can be in both states at the same time." Any measurement of the state of a qubit, like any measurement of the state of a classical bit, can yield only one or the other of two and only two possible states. Because a qubit is a quantum mechanical system describable by a wave function, however, the two exclusive possible outcome states for a state measurement performed on a qubit typically will depend on measurement details, which is not the case for a classical bit. Suppose for instance that our qubit is a spin $\frac{1}{2}$ particle, one of many conceivable physical realizations of a qubit in a practical quantum computer.[51] Then a measurement of the component of the particle's spin along the z direction can yield the results $+\frac{1}{2}$ and $-\frac{1}{2}$ only, to which correspond the orthogonal wave functions commonly denoted by $|+z\rangle$ and $|-z\rangle$, respectively. Similarly, if a measurement of the component of spin along the y direction is performed on a particle which has been found to have spin $+\frac{1}{2}$ along the z direction, the only possible results again are $+\frac{1}{2}$ and $-\frac{1}{2}$ only, to which correspond orthogonal wave functions $|+y\rangle$ and $|-y\rangle$ respectively. But neither of the wave functions $|+y\rangle$ and $|-y\rangle$ is identical with the wave function $|+z\rangle$ or $|-z\rangle$. Rather each of the wave functions $|+z\rangle$ and $|-z\rangle$ is a known linear combination of the wave functions $|+y\rangle$ and $|-y\rangle$ and vice versa.[52,53]

1. The computational basis: Quantum computer wave functions

To enable convenient employment of a qubit for computational purposes, namely, in order that the two possible outcomes of state measurements on the qubit be consistently interpretable as corresponding to the binary integers 0 and 1 respectively, it is necessary to assume that the qubit state measurement always will be performed in the same way, for example, with a Stern-Gerlach apparatus always lined up along the positive z direction if the qubit is a spin $\frac{1}{2}$ particle. With this assumption the pair of orthogonal wave functions describing the two possible qubit state measurement outcomes customarily are denoted by $|0\rangle$ and $|1\rangle$. These wave functions comprise the so-called *computational basis*, and are interpretable respectively as corresponding to the binary integers 0 and 1. The wave function Ψ describing any arbitrary state of the qubit, which is a linear superposition of any pair of orthogonal wave functions, typically is expanded in terms of $|0\rangle$ and $|1\rangle$ only:

$$\Psi=\mu|0\rangle+\nu|1\rangle, \tag{15}$$

where μ and ν are a pair of complex numbers satisfying

$$|\mu|^2+|\nu|^2=1. \tag{16}$$

Equation (16), which expresses that Ψ is normalized to unity (as are $|0\rangle$, $|1\rangle$ and all other wave functions discussed below), permits the interpretation that $|\mu|^2$ is the probability that a state measurement will yield the outcome corresponding to

0, and $|\nu|^2$ is the probability that the same measurement will yield the outcome corresponding to 1.

A quantum computer is a collection of qubits, and thus it is a quantum mechanical system whose state must be describable by a normalized wave function. Consider, in particular, a computer composed of just two qubits, labeled by A and B. There now are at most $2\times2=4$ possible different outcomes of state measurements on the pair of qubits A, B (whether performed simultaneously or successively). Consequently, the wave function Ψ of this computer must be a linear superposition of at most four orthogonal two-qubit basis wave functions, which as Mermin[1] fully discusses can be taken to be the computational basis products $|0\rangle_B|0\rangle_A \equiv|00\rangle$, $|0\rangle_B|1\rangle_A\equiv|01\rangle$, $|1\rangle_B|0\rangle_A\equiv|10\rangle$, and $|1\rangle_B|1\rangle_A \equiv|11\rangle$. In other words, the most general two-qubit computer wave function has the form

$$\Psi=\gamma_{00}|00\rangle+\gamma_{01}|01\rangle+\gamma_{10}|10\rangle+\gamma_{11}|11\rangle, \tag{17}$$

where in the computational basis wave functions $|00\rangle$, etc., it is understood that the two binary digits reading from left to right correspond to the outcomes of state measurements on qubits B and A, respectively, and where the associated amplitudes γ_{00}, etc., are complex numbers satisfying

$$|\gamma_{00}|^2+|\gamma_{01}|^2+|\gamma_{10}|^2+|\gamma_{11}|^2=1. \tag{18}$$

The digit pairs 00, 01, 10, and 11 indexing the computational basis wave functions appearing in Eq. (17) are the binary system representations of the decimal system integers 0 through 3, with the proviso that each such binary representation is to consist of no fewer than two digits. Thus Eq. (17) can be rewritten as

$$\Psi=\gamma_0|0\rangle+\gamma_1|1\rangle+\gamma_2|2\rangle+\gamma_3|3\rangle, \tag{19}$$

where the basis wave functions $|i\rangle$ and associated amplitudes γ_i, $i=1-3$, are merely relabelings, respectively, of the basis wave functions $|00\rangle$, $|01\rangle$, etc., and of the amplitudes γ_{00}, γ_{01}, etc.

2. Wave function collapse and the Born rule

In Eqs. (17) and (18) each $|\gamma_{\beta\alpha}|^2$ is the probability that measurements on the qubit pair A, B in the two-qubit state described by Ψ will yield state $|\alpha\rangle_A$ for qubit A and state $|\beta\rangle_B$ for qubit B, where α and β can have the values 0 and 1 only. It is conceivable, however, that the observer will seek to measure the state of qubit A only, without any attempt to ascertain the state of qubit B. In this event $|\gamma_{00}|^2+|\gamma_{10}|^2$ is the probability of finding A in state $|0\rangle_A$, and $|\gamma_{01}|^2 +|\gamma_{11}|^2$ is the probability of finding A in state $|1\rangle_A$. If a measurement on qubit A is performed, and A actually is found to be in state $|0\rangle_A$, then the original wave function Ψ of Eq. (17) is said to have been *reduced* or *collapsed* by the measurement into the new wave function $\Psi'=\Psi_B|0\rangle_A$, where the one-qubit wave function Ψ_B for qubit B is

$$\Psi_B=[|\gamma_{00}|^2+|\gamma_{10}|^2]^{-1/2}[\gamma_{00}|0\rangle_B+\gamma_{10}|1\rangle_B]. \tag{20}$$

Equation (20) is in accordance with the *Born rule*, which Mermin[1] discusses. The normalizing factor $[|\gamma_{00}|^2 +|\gamma_{10}|^2]^{-1/2}$ in Eq. (20) is needed to ensure that Ψ' and Ψ_B are normalized wave functions, that is, that in Ψ' and in the state of B described by the one-qubit wave function Ψ_B, the individual probabilities of finding qubit B in state $|0\rangle_B$ and in state $|1\rangle_B$ sum to unity. Note that the square of the coefficient of $|\beta\rangle_B$ in Eq. (20), which represents the probability of finding qubit B in the state $|\beta\rangle_B$ *knowing* that a measurement on qubit A in the two-qubit state described by Ψ of Eq. (17) already has yielded $|0\rangle_A$, differs from $|\gamma_{\beta0}|^2$ representing the probability, *without any such knowledge*, that measurements on the qubit pair A, B in the two-qubit state described by Ψ will find qubit A in state $|0\rangle_A$ and qubit B in state $|\beta\rangle_B$. The modification of Eq. (20) appropriate to the circumstance that A actually had been found in state $|1\rangle_A$ rather than in state $|0\rangle_A$ is obvious. Equally obvious [starting again with the two-qubit system in the state described by Ψ of Eq. (17)] is that (i) the probability of finding qubit B in state $|\beta\rangle_B$ (β is either 0 or 1) without any attempt to ascertain the state of qubit A is $\Sigma_\alpha|\gamma_{\beta\alpha}|^2$, with the sum of these probabilities $=\Sigma_\beta\Sigma_\alpha|\gamma_{\beta\alpha}|^2=1$; and ($ii$) if B actually is found in state $|\beta\rangle_B$ (β either 0 or 1), the original Ψ collapses into the wave function $\Psi_A|\beta\rangle_B$, where

$$\Psi_A=[|\gamma_{\beta0}|^2+|\gamma_{\beta1}|^2]^{-1/2}[\gamma_{\beta0}|0\rangle_A+\gamma_{\beta1}|1\rangle_A]. \tag{21}$$

These considerations are immediately extensible to larger quantum computers, composed of $g\geqslant2$ qubits. Because a state measurement on any given qubit can have at most two different outcomes, state measurements on the entire collection of qubits comprising a g-qubit quantum computer can have at most 2^g different outcomes. Correspondingly, the wave function Ψ describing any state of a g-qubit quantum computer is a linear superposition of at most 2^g orthogonal g-qubit basis wave functions. If we index these g qubits by k running from 1 to g, then the 2^g computational basis wave functions for the computer can be taken to be

$$\begin{aligned}&|0\rangle_g|0\rangle_{g-1}\cdots|0\rangle_2|0\rangle_1\equiv|00\cdots00\rangle,\\ &|0\rangle_g|0\rangle_{g-1}\cdots|0\rangle_2|1\rangle_1\equiv|00\cdots01\rangle,\ldots,\\ &|1\rangle_g|0\rangle_{g-1}\cdots|0\rangle_2|1\rangle_1\equiv|10\cdots01\rangle,\end{aligned} \tag{22}$$

etc., and in analogy to Eq. (19) the most general g-qubit quantum computer wave function can be expressed as

$$\Psi=\sum_{i=0}^{2^g-1}\gamma_i|i\rangle, \tag{23}$$

with

$$\sum_{i=0}^{2^g-1}|\gamma_i|^2=1. \tag{24}$$

In Eqs. (23) and (24) the integers i are conveniently written in the decimal system, as in Eq. (19); the binary system representation of each i consists of no fewer than g digits. Each computational basis wave function $|i\rangle$ represents a g-qubit state, where for every k, $1\leqslant k\leqslant g$, the outcome (0 or 1) of a state measurement on the kth qubit equals the kth digit (reading from right to left) in the binary system representation of i; $|\gamma_i|^2$ is the probability that when the computer is in the state described by the wave function Ψ of Eq. (23), state measurements on the collection of g qubits will have the same outcomes as if the computer wave function is simply $|i\rangle$. Moreover, if, while the computer is in the state described by Ψ of Eq. (23), the computer operator were to measure, for example, the states of qubits 1, 2 and g and obtain the outcomes $|1\rangle_1$, $|0\rangle_2$, and $|1\rangle_g$, respectively, these measurements would collapse Ψ into the wave function $\Psi_M = \Psi_{g-3}[|1\rangle_1 |0\rangle_2 |1\rangle g]$, where the $(g-3)$-qubit wave function,

$$\Psi_{g-3} = \left[\sum_j |\gamma_j|^2\right]^{-1/2} \sum_j \gamma_j |j\rangle, \tag{25}$$

describes the state of the remaining qubits 3,4,...,$(g-1)$ knowing that state measurements on qubits 1, 2, and g in the g-qubit system described by Ψ had yielded the outcomes $|1\rangle_1$, $|0\rangle_2$, and $|1\rangle_g$ respectively. In addition, j in Eq. (25) runs over all integers whose g-digit binary representations begin with 1 and end with 01 (now reading from left to right).

3. Operations on quantum computers: Unitarity

It can be assumed that the g-qubit quantum computer wave function Ψ of Eq. (23), like the wave function of any other quantum mechanical system, evolves in accordance with the nonrelativistic time-dependent Schrödinger equation,

$$\frac{\partial \Psi}{\partial t} = \frac{ih}{2\pi} \hat{H} \Psi, \tag{26}$$

where h is Planck's constant and the Hamiltonian $\hat{H}$, which may be time dependent, is an appropriate Hermitian operator capable of acting on the various computational basis wave functions $|i\rangle$ appearing in Eq. (23). In this circumstance the wave function $\Psi(t)$ at any time $t \geqslant 0$ is related to the wave function $\Psi(0)$ at $t=0$ by

$$\Psi(t) = \hat{U}\Psi(0), \tag{27}$$

where, because $\hat{H}$ is Hermitian, $\hat{U} \equiv \hat{U}(t)$ is a linear normalization-conserving operator,[54] that is, a unitary operator.[55] Whatever the physical realizations of the individual qubits comprising the quantum computer may be, the computer's utility as a computational tool depends on the ability (of the person performing the computation) to control the evolution of its wave function. But this desired controlled evolution, which generally requires modifying the environments of the individual qubits (for example, when the qubits are spin $\frac{1}{2}$ particles, rotating the individual magnetic fields acting on those particles), is necessarily an evolution of Ψ under Eq. (26). Thus the desired controlled evolution also is described by Eq. (27), that is, it involves a unitary operation on the initial wave function $\Psi(0)$.

Accordingly each planned operation in the sequence of operations constituting any proposed quantum computing algorithm, for example, Shor's algorithm, must be a unitary operation. Postulated nonunitary operations on a quantum computer, no matter how seemingly attractive, are irrelevant and thus of no interest for the use of the computer as a computational tool, because no nonunitary operation will be attainable with any actual physical realizations of the qubits comprising the computer. Therefore, note that each of the quantum computing operations in Shor's algorithm is unitary.

The impossibility of constructing a physical realization of any nonunitary operation does not imply that every conceivable unitary operation on a quantum computer can be physically realized. Furthermore, if the computer is composed of a large number of qubits, for example, thousands of qubits (as is likely in practical applications of Shor's algorithm), the prospect of actually constructing a physical realization of any nontrivial unitary operation $\hat{U}$ on so large a collection of qubits seems hopeless at first sight, even if there is reason to believe that a physical realization of $\hat{U}$ must exist. Fortunately, however, and absolutely crucial for the practical potential of quantum computation, it can be proved that every conceivable unitary operation on an arbitrarily large g-qubit quantum computer, even an operation involving simultaneous modifications of the environments of all $g \geqslant 2$ qubits, can be reproduced via an appropriate sequence of unitary one-qubit and two-qubit operations only.[56] Moreover, numerous methods based solely on known physics for achieving physical realizations of these basic unitary one-qubit and two-qubit operations (also known as *universal quantum gates*[56]) have been proposed,[51] although admittedly the actual implementation of many of these potential physical realizations may prove to be difficult in practice.

For the purposes of this paper it is reasonable to assume that quantum computers consisting of arbitrarily large assemblages of qubits, capable of performing any desired computational algorithm that can be formulated in terms of unitary operations, will eventually be constructed. Given this assumption, a measure of the quantum computational effort required to perform any given algorithm, indeed the only obvious measure, is the number of universal quantum gates that must be strung together to perform the algorithm on a quantum computer. In essence the universal quantum gate operations play the role, for quantum computation, that the bit operations referred to in connection with Eq. (7) play for classical computation.

We now are able to make precise the meaning of the oft-repeated assertion that the Shor algorithm enables a quantum computer to factor large key numbers $N=pq$ with far less computational effort than using a classical computer requires. In particular, with a quantum computer using Shor's algorithm, the number ν_q of universal quantum gates required to determine an order r that will enable factorization of a large $N=pq$ via Eq. (11) has been estimated[3,4] to be

$$\nu_q(N) = O[(\ln N)^2 (\ln \ln N)(\ln \ln \ln N)] = O[L^2(\log_2 L) \times (\log_2 \log_2 L)], \tag{28}$$

where $L = \log_2 N$ as in Eq. (7). The symbol O, denoting *order of*, implies[57] that there exists a constant K such that for sufficiently large N

$$\nu_q(N) \leqslant K[L^2(\log_2 L)(\log_2 \log_2 L)]. \tag{29}$$

In connection with Eqs. (28) and (29) it is useful to recognize that for large N the number of qubits required to represent N is essentially L. To be precise, for any real number $x \geqslant 1$, let $[x]$ denote the largest integer less than or equal to x. Then it is easily seen that the number of qubits needed to represent any N is $[\log_2 N]+1$, which for large N differs negligibly from L.

Our discussion has overlooked a needed refinement to Eq. (28), as well as the fact that in practice the actual factorization of N using Shor's algorithm requires computational operations (for example, classical computer gcd calculations) beyond the universal quantum gate operations whose number is estimated in Eq. (28) (see Sec. IV). However, the discussion in Sec. IV implies that for the purposes of this paper neither the aforementioned refinement nor such neglected computational operations negate the utility of Eq. (29) as a measure of the computational effort required to factor a large $N = pq$ with a quantum computer using Shor's algorithm. Therefore, a comparison of Eqs. (7) and (29) correctly quantifies the reduction in the computational effort required to factor a large N that is achievable with a quantum computer. That is, according to Eq. (7) the number of elemental computer operations needed to accomplish the factorization of N with a classical computer increases faster than any power of $L = \log_2 N$. In contrast, the needed number of elemental computer operations using a quantum computer increases only a little more rapidly than L^2 (indeed surely less rapidly than L^3) according to Eq. (29).

To illustrate the practical import of this reduction, let us repeat the numerical exercise presented immediately below Eq. (7), only this time for a quantum computer. For any sufficiently large quantum computer, that is, for any quantum computer composed of sufficiently many qubits to handle the Shor algorithm determination of r for all relevant N, the time $\tau_q(N)$ needed to complete the factorization of a large N should be approximately proportional to $\nu_q(N)$ given by Eq. (29), irrespective of the value of K appropriate for that computer. Suppose we were able to construct a quantum computer which, like the classical computer we hypothesized previously, could factor RSA-309 in two weeks time. Then this same quantum computer should be able to factor RSA-617 in no more than about nine weeks, in contrast to the 60 million years for the classical computer.

Moreover, it is reasonable to believe that a sufficiently large quantum computer will be able to factor RSA-309 in about two weeks or $\approx 1.2 \times 10^6$ s. For RSA-309, that is, for $L = 1024$, the value of ν_q from Eq. (29) is only 3.5×10^9 even assuming K is as large as 100, which seems doubtful. Therefore, to factor RSA-309 in two weeks, the average time for performing a quantum gate operation need be no faster than about 300 μs, which should be no problem for quantum computer elements, whether operating on atomic, molecular, or photonic scales. In short, once sufficiently large quantum computers become available, Bob no longer will be justifiably confident that he can maintain the security of Alice's RSA-coded messages to him, merely by increases of his proclaimed key number size, in the face of anticipated improvements in quantum computer capabilities.

Before finishing this discussion of operations on quantum computers, it is important to note that wave function collapsing measurements on any part of a quantum computer, though normalization conserving by virtue of the Born rule,[1] are not—strictly speaking—quantum computing operations of the sort discussed earlier. In particular, let Ψ_M denote the wave function defined immediately preceding Eq. (25). Then, as Mermin has discussed,[1] because both Ψ of Eq. (23) and Ψ_M are normalized wave functions expressible as linear superpositions of the very same set of 2^g orthogonal computational basis wave functions, there must exist a unitary operator $\hat{U}_M$ such that

$$\Psi_M = \hat{U}_M \Psi. \tag{30}$$

Because Ψ can be thought of as $\Psi(0)$, the computer wave function at time $t=0$ when the measurement operation began, and Ψ_M can be thought of as $\Psi(t)$, the computer wave function at time $t>0$ when the measurement has been completed, Eq. (30) appears to have the same form as Eq. (27). The subtle difference is that whereas in Eq. (27) we have been considering unitary operators which are predictably controllable [that is, which during each step of the computational algorithm will cause the computer wave function $\Psi(0)$ to evolve into some desired $\Psi(t)$], $\hat{U}_M$ of Eq. (30) generally is not predictably controllable. Rather the $\hat{U}_M$ we obtain as a result of the measurement generally is only one of many possible $\hat{U}_M$, whose likelihoods of turning up in the actual measurement operation we have performed depend on the values of the coefficients γ_i in Eq. (23). Only after we observe the measurement outcome can we decide which of the many possible $\hat{U}_M$ actually has been obtained. Correspondingly, there generally is no way before the measurement operation to introduce a sequence of universal quantum gates that will reproduce the unitary operator $\hat{U}_M$ of Eq. (30) that actually is attained.

C. The operations constituting Shor's algorithm

Shor's original formulation[3] of his algorithm has been given an admirably readable (by nonspecialists) step-by-step prescription by Williams and Clearwater,[58] which my presentation will follow closely, but also will expand on and illustrate. Each subheading in this section briefly describes one of the eight steps.

1. Determine the minimum computer size required: Divide the qubits into two registers

Shor's algorithm seeks to accurately discern the periodicity with period r manifested by the sequence f_j, $j = 1,2,3,\ldots$, obtained from Eq. (8) for some chosen n. To do so, the algorithm must operate on sequences that are many periods in length, much as in conventional classical Fourier transformation. In practice the order r may attain its maximum possible value $(p-1)(q-1)/2$, which for large N is likely to be only slightly smaller than half of $N = pq$ (see Appendix B). For our example $N = 55$, the order r equals its maximum allowed value 20 for fully 16 of the 40 integers $n < 55$ that are coprime to 55, including n as small as 2 and as large as 53. Consequently, the accurate determination of r using Shor's algorithm generally requires the use of powers $j \gg N$ in Eq. (8). Shor recommends (in effect) that the maximum power $j = j_{\max}$ employed be no less than N^2, a recommendation this paper accepts.[3] Williams and Clearwater[58] recommend $j_{\max}$ be even greater, namely at least $2N^2$. Thus, following Shor, the quantum computer being employed to determine r via Shor's algorithm must contain at least enough qubits to represent powers j up to $j_{\max} = N^2$. The minimum number of qubits needed to represent the integer N^2 is $[\log_2 N^2] + 1$. Thus, the quantum computer should contain a set of $y = [\log_2 N^2] + 1$ qubits, which will comprise register Y. In addition, the computer must contain a second set of qubits, here termed register Z, capable of storing the computed values of f_j, which can be as large as $N-1$. The size of this register will be taken to be its minimum possible value $z = [\log_2(N-1)] + 1$ qubits. Note that because N is known to be the product of a pair of odd primes and thus is not a power of 2, it follows that $2^{y-1} < N^2 < 2^y$.

For large N the difference between $[\log_2(N^2)] + 1$ and $2L = \log_2(N^2)$, like the difference between $[\log_2(N-1)]$ and L or between L and $L+1$, is negligible for the purpose of estimating the computational effort required to accomplish the various individual steps in Shor's algorithm. Thus, in any subsequent estimates y can be replaced by $2L$ (ignoring the fact that $2L$ need not be an integer). This is the same replacement for y employed[4] to obtain Eq. (28). For the purpose of such estimates the difference between $[\log_2(N^2)]\cong 2L$ and $[\log_2(2N^2)]\cong 2L+1$ also is negligible, meaning that the estimated computational effort required to accomplish the various individual steps constituting Shor's algorithm do not depend significantly on whether we prefer the Shor or the Williams-Clearwater estimates of the required $j_{\max}$. Furthermore, we now can conclude that unless for large N the actually required value of $j_{\max}$ is very much smaller than Shor anticipates, determining r and thereby factoring a large N will require a quantum computer not less than about $3L$ qubits in size. In other words, factoring a key number of the presently recommended size RSA-309 corresponding to 1024 binary digits (recall Sec. II D) seemingly would require a quantum computer of at least 3072 qubits in size; factoring RSA-617 would require a quantum computer of more than 6000 qubits in size.

2. Load the first register with the integers less than or equal to 2^y-1

After ordering and indexing the y qubits in register Y as discussed in connection with Eqs. (22)–(25), the complete set of computational basis wave functions for those qubits can be written as $|j\rangle_Y$, where the subscript indicates that we are writing wave functions for register Y; j is an integer, $0 \leqslant j \leqslant 2^y-1$, which will be written in decimal notation. When register Y is in the state described by the basis wave function $|j\rangle_Y$, the binary digit representation of j immediately reveals the one-qubit basis state, $|0\rangle_k$ or $|1\rangle_k$, of each of the qubits in register Y. It is understood that qubit k $(1\leqslant k\leqslant y)$, whose basis states are identified by the subscript k, corresponds to the kth digit, reading from right to left, in the binary system representation of j. The computational basis wave functions for register Z similarly are denoted by $|i\rangle_Z$, where $0\leqslant i \leqslant 2^z-1$. It is postulated that initially every one of the $y+z$ qubits constituting the quantum computer can be set into its own one-qubit $|0\rangle$ basis state, that is, the initial wave function of the entire quantum computer is $\Psi_C^{(0)}=|0\rangle_Y|0\rangle_Z$, where the subscript C denotes the wave function of the entire computer. Proceeding with the algorithm requires transforming the initial register Y wave function $\Psi_Y^{(0)}\equiv|0\rangle_Y$ to its second step form,

$$\Psi_Y^{2S}=2^{-y/2}\sum_{j=0}^{2^y-1}|j\rangle_Y, \tag{31}$$

that is, requires replacing the initial $|0\rangle_Y$ by the sum on the right side of Eq. (31), wherein a measurement of the state of register Y has an equal chance of yielding any of the integers between 0 and 2^y-1 inclusive. There are 2^y independent $|j\rangle_Y$ on the right side of Eq. (31). Thus, the factor $2^{-y/2}$ guarantees Ψ_Y^{2S} is normalized. Because we know $N^2\leqslant 2^y-1$, the sum in Eq. (31) includes every j less than or equal to Shor's recommended $j_{\max}=N^2$.

The transformation of $|0\rangle_Y$ to Ψ_Y^{2S} of Eq. (31) is accomplished by the use of the one-qubit operation $\hat{U}_H$ known as a *Hadamard* transformation, which is defined[59] so that the results of the Hadamard operation on the one-qubit basis state wave functions $|0\rangle$ and $|1\rangle$ are

$$\hat{U}_H|0\rangle=\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle),\quad \hat{U}_H|1\rangle=\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle). \tag{32}$$

$\hat{U}_H$ is known to be unitary.[59] The factor $1/\sqrt{2}$ in Eq. (32) enables $\hat{U}_H$ to preserve normalization, as we know a unitary operation must.[55] Denote the operation that performs the Hadamard on qubit k alone, without affecting any other qubits, by $\hat{U}_{Hk}$. Next consider the result of operating with $\hat{U}_{Hk}$ on a computational basis wave function $|j\rangle_Y$ for which the kth digit (reading from right to left) in the binary expansion of j is 0 (not 1), meaning that the product of one-qubit basis wave functions constituting $|j\rangle_Y$ includes the factor $|0\rangle_k$ (not $|1\rangle_k$). To obtain this desired result, we need only put the subscript k on every one of the basis wave functions in Eq. (32). Moreover, because our present $|j\rangle_Y$ contains no $|1\rangle_k$ basis state, we are here concerned only with the first equality in Eq. (32). It follows that, except for the factor $1/\sqrt{2}$, the operation $\hat{U}_{Hk}$ on our present $|j\rangle_Y$ merely replaces $|0\rangle_k$ in $|j\rangle_Y$ by $|0\rangle_k+|1\rangle_k$, while leaving $|j\rangle_Y$ otherwise unchanged. In the binary expansion of the integer j, however, changing the kth digit from 0 to 1 (always reading from right to left) produces the binary expansion of the integer $j+2^{k-1}$. Therefore, when $|j\rangle_Y$ contains no $|1\rangle_k$ basis state,

$$\hat{U}_{Hk}|j\rangle_Y=\frac{1}{\sqrt{2}}[|j\rangle_Y+|j+2^{k-1}\rangle_Y]. \tag{33}$$

Now perform the y operations $\hat{U}_{H1}$, $\hat{U}_{H2}$, $\hat{U}_{H3},\ldots,\hat{U}_{Hy}$ sequentially (first $\hat{U}_{H1}$) on the initial register Y wave function $|0\rangle_Y\equiv\Psi_Y^{(0)}$. We know $|0\rangle_Y$ contains no $|1\rangle_i$ factors for any i, $1\leqslant i\leqslant y$. Thus we surely can employ Eq. (33) for the first of these operations to obtain

$$\begin{aligned}\Psi_Y^{(1)}=\hat{U}_{H1}|0\rangle&=\frac{1}{\sqrt{2}}[|0\rangle_Y+|0+2^0\rangle_Y]\\ &=\frac{1}{\sqrt{2}}[|0\rangle_Y+|1\rangle_Y]\\ &=\frac{1}{\sqrt{2}}\sum_{j=0}^{1}|j\rangle_Y.\end{aligned} \tag{34}$$

Because $\hat{U}_{H1}$ has been defined so that it performs the Hadamard operation on qubit 1 only, the wave function $\Psi_Y^{(1)}$ (like $\Psi_Y^{(0)}$) does not contain the factor $|1\rangle_2$, as is directly evidenced by the fact that both the integers 0 and 1 on the right side of Eq. (34) are less than 2. Consequently, we also can employ Eq. (33) for the second of these sequential operations, thereby finding for $\hat{U}_{H2}\hat{U}_{H1}|0\rangle=\hat{U}_{H2}\Psi_Y^{(1)}\equiv\Psi_Y^{(2)}$,

$$\begin{aligned}\Psi_Y^{(2)}=\frac{1}{\sqrt{2}}\hat{U}_{H2}[|0\rangle_Y+|1\rangle_Y]&=\frac{1}{2}[[|0\rangle_Y+|0+2\rangle_Y]\\ &\quad+[|1\rangle_Y+|1+2\rangle_Y]]\\ &=\frac{1}{2}\sum_{j=0}^{3}|j\rangle_Y.\end{aligned} \tag{35}$$

Because every one of the integers on the right side of Eq. (35) is less than $4=2^2$, $\Psi_Y^{(2)}$ surely does not contain the factor $|1\rangle_3$, permitting use of Eq. (33) to evaluate $\hat{U}_{H3}\Psi_Y^{(2)}$. Proceeding in this fashion, it is readily seen that the result of the full sequence of Hadamard operations on $|0\rangle_Y$ is

$$\Psi_Y^{(y)} = \hat{U}_{Hy}\hat{U}_{H(y-1)}\cdots\hat{U}_{H2}\hat{U}_{H1}|0\rangle = 2^{-y/2}\sum_{j=0}^{2^y-1}|j\rangle_Y. \tag{36}$$

The right side of Eq. (36) is the desired Ψ_Y^{2S} of Eq. (31). It is generally agreed[60] that the above-defined Hadamard one-qubit operations are universal quantum gates. Accordingly, accomplishing this second step transformation of the initial $|0\rangle_Y$ to Ψ_Y^{2S} requires no more than $y=2L$ universal quantum gates.

3. Select an n and for each j in register Y, place the remainder $f_j \equiv n^j$ (mod N) into register Z

After this just completed second step, the wave function of the entire computer is $\Psi_C^{2S} = \Psi_Y^{2S}|0\rangle_Z$, meaning that after completion of the first stage, a state measurement on register Z still is guaranteed to yield the integer 0 only, irrespective of what value of j came from a simultaneous state measurement on register Y. For the next step an n coprime to N is required. As Appendix B explains, such an n can be obtained, with probability essentially indistinguishable from unity, simply by choosing an arbitrary integer i, either in the range $1<i<N$ or in the range $1<i<2^y-1$. Whether any selected integer i actually is coprime to N readily can be tested by calculating the gcd of i and N using a classical computer (as discussed in Appendix D 1), but the probability such a randomly chosen i will not be coprime to N is so small that the effort of computing this gcd does not seem worthwhile. If the selected i is not coprime to N, this fact will become apparent when it is realized that the value of the supposed order r, inferred as in step 7, does not satisfy Eq. (10). Appendix B explains that no integer r can satisfy Eq. (10) when the integer n in Eq. (10) is not coprime to N. In this event it will be necessary to repeat steps 2–7 after choosing a different i, which will almost certainly be coprime to N. Such repetitions often are required even when the chosen i is coprime to N (see Sec. III C 8).

Assuming now an $i=n$ coprime to N actually has been selected, the next step of the algorithm transforms Ψ_C^{2S} to its third step form

$$\Psi_C^{3S} = 2^{-y/2}\sum_{j=0}^{2^y-1}|j\rangle_Y|f_j\rangle_Z, \tag{37}$$

where f_j is defined by Eq. (8). With the computer wave function Ψ_C^{3S} of Eq. (37), the result of a state measurement on the collection of qubits in register Z must yield one of the remainder integers $1\le f_j\le N-1$ prescribed by Eq. (8). Moreover, because of the periodicity of f_j demonstrated in Appendix A, every one of the f_j in Eq. (37) must equal one of the (all necessarily different) $f_1, f_2,\dots,f_r=f_0=1$. No other integers can result from a state measurement on register Z after completion of the second stage of the algorithm. In particular, because n is coprime to N by definition, such a measurement now cannot possibly yield the previously (at completion of the first step) assured result 0.

I shall not detail the operation that transforms Ψ_C^{2S} to Ψ_C^{3S} of Eq. (37). The operation is fully discussed by Shor,[3] who showed that it is unitary. The number of universal quantum gates required to perform this unitary operation is[3,4] $O[L^2(\log_2 L)(\log_2\log_2 L)]$. Let us illustrate Eq. (37) when $N=55$ and $n=16$. In these circumstances, as discussed in Sec. III A, $r=5$ and the sequence f_j (now starting with $j=0$) is 1, 16, 36, 26, 31, 1, 16, 36, 26, 31, 1, 16, 36,... . Accordingly, in this illustrative case Eq. (37) is

$$\begin{aligned}\Psi_C^{3S} = 2^{-y/2}[&|0\rangle_Y|1\rangle_Z + |1\rangle_Y|16\rangle_Z + |2\rangle_Y|36\rangle_Z \\ &+|3\rangle_Y|26\rangle_Z + |4\rangle_Y|31\rangle_Z + |5\rangle_Y|1\rangle_Z + |6\rangle_Y|16\rangle_Z \\ &+|7\rangle_Y|36\rangle_Z + |8\rangle_Y|26\rangle_Z + |9\rangle_Y|31\rangle_Z + |10\rangle_Y|1\rangle_Z \\ &+|11\rangle_Y|16\rangle_Z + |12\rangle_Y|36\rangle_Z + \cdots + |2^y \\ &-1\rangle_Y|f_{2^y-1}\rangle_Z].\end{aligned} \tag{38}$$

As Eq. (38) illustrates, the sequence of register Z basis wave functions $|1\rangle_Z, |f_1\rangle_Z, |f_2\rangle_Z,\dots,|f_{2^y-1}\rangle_Z$ in Eq. (37) manifests the same periodicity with r as does its originating sequence f_j, $0\le j\le 2^y-1$.

4. Measure the state of register Z

The entire computer now is in the state represented by Ψ_C^{3S} of Eq. (37). The objective of the next three steps in the algorithm is to extract the value of r from the just discussed periodicity of Ψ_C^{3S}. Note that although we know Ψ_C^{3S} has the form given in Eq. (37), until we begin making measurements on register Z, we can have no idea of what values of f_j actually are appearing in Eq. (37). Moreover, the wave function collapse discussed in Sec. III B 2 means that any single measurement on register Z, though it surely will reveal one of the values of f_j appearing in Eq. (37), will automatically destroy all information about the other values of f_j. Nevertheless, the next step in the algorithm is to measure the state of register Z. Suppose this register Z measurement on the computer in the state represented by Eq. (37) yields the particular value f_k (of the r possible values $f_0=1$, f_1, $f_2,\dots,f_{r-1}$). Then after the measurement the wave function of register Y takes its fourth step form

$$\Psi_Y^{4S} = Q^{-1/2}\sum_{b=0}^{Q-1}|k+br\rangle_Y, \tag{39}$$

where Eq. (39) has retained those and only those $|j\rangle_Y$ in Eq. (37) that are multiplied by $|f_k\rangle_Z$. Q equals the number of terms in Eq. (37) containing the factor $|f_k\rangle_Z$; the factor $Q^{-1/2}$ is necessary to guarantee the wave function Ψ_Y^{4S} of Eq. (39) is normalized, consistent with the Born rule.[1]

To help comprehend the structure of Eq. (39) and to see how the magnitude of Q is estimated, let us return to our $N=55$, $n=16$, $r=5$ example. Suppose the result of the register Z measurement on the computer in the state represented by Eq. (38) was $f_j=36$. Then after the measurement, the wave function of register Y in this fourth stage of the operation of the algorithm was

$$\Psi_Y^{4S} = Q^{-1/2}[|2\rangle_Y + |7\rangle_Y + |12\rangle_Y + \cdots + |2+5(Q-1)\rangle_Y]. \tag{40}$$

Evidently the measurement has shifted the dependence on r, from the periodicity with r of the sequence $|f_j\rangle_Z$ in Eq. (37), to the periodicity of an arithmetic progression (with the common difference r) of the integers $j=k+br$ indexing the computational basis wave functions $|j\rangle_Y$ appearing in Eq. (39). It is evident from Eq. (40) that the value of Q in Eq. (39) is determined by the condition that $k+r(Q-1)$ cannot exceed 2^y-1, the largest j appearing in Eq. (37). Because $0\leqslant k<r$ and Q is an integer by definition, this condition implies

$$Q=[r^{-1}(2^y-1-k)]+1, \tag{41}$$

with $[x]$ denoting the largest integer less than or equal to x. We see that unless $2^y/r$ is an integer, Q in Eq. (39) either equals $[2^y/r]$ or $[2^y/r]+1$, depending on the value of k; for the large N cases of interest here, either of these two possible values of Q is well approximated by $2^y/r$, because $r<N/2$ (see Appendix B) whereas $2^y>N^2$ (as noted in Sec. III C 1). When $2^y/r$ is an integer, however (recall the illustrative f_j sequence $N=55$, $n=12$ discussed in Sec. III A), Eq. (41) makes Q exactly equal to $2^y/r$ for every allowed value of k.

5. Perform a quantum Fourier transform operation on the register Y wave function

The desired r finally is extracted from Ψ_Y^{4S} of Eq. (39) via a quantum Fourier transform operation $\hat{U}_{FT}$. The operation $\hat{U}_{FT}$ transforms any state $|j\rangle_Y$ in register Y to

$$\hat{U}_{FT}|j\rangle_Y=2^{-y/2}\sum_{c=0}^{2^y-1}e^{2\pi ijc/2^y}|c\rangle_Y. \tag{42}$$

After the operation $\hat{U}_{FT}$, therefore, the wave function for register Y takes its fifth step form

$$\Psi_Y^{5S}=\hat{U}_{FT}\Psi_Y^{4S}=(2^yQ)^{-1/2}\sum_{c=0}^{2^y-1}\sum_{b=0}^{Q-1}e^{2\pi i(k+br)c/2^y}|c\rangle_Y. \tag{43}$$

It has been shown[4,61] that $\hat{U}_{FT}$ can be written as a product of universal quantum gates, implying that $\hat{U}_{FT}$ is unitary, as required. The number of gates required is $O(L^2)$.[4,61]

In Eq. (43) the coefficient of any given $|c\rangle_Y$ is a geometric series and can be trivially summed to give

$$\Psi_Y^{5S}=(2^yQ)^{-1/2}\sum_{c=0}^{2^y-1}e^{2\pi ikc/2^y}\frac{1-e^{2\pi ircQ/2^y}}{1-e^{2\pi irc/2^y}}|c\rangle_Y$$
$$=(2^yQ)^{-1/2}\sum_{c=0}^{2^y-1}e^{2\pi ikc/2^y}e^{\pi irc(Q-1)/2^y}\times\frac{\sin(\pi rcQ/2^y)}{\sin(\pi rc/2^y)}|c\rangle_Y. \tag{44}$$

6. Measure the state of the Y register

This measurement will find the Y register in a particular state $|c\rangle_Y$. The probability P_c of finding the state $|c\rangle_Y$ is given by the square of the absolute value of the coefficient of $|c\rangle_Y$ in Eq. (44), namely,

$$P_c=(2^yQ)^{-1}\frac{\sin^2(\pi rcQ/2^y)}{\sin^2(\pi rc/2^y)}. \tag{45}$$

It is worth remarking that because in step 5 the operation $\hat{U}_{FT}$ does not involve the Z register, the same Eq. (45) for the probability of finding the Y register in the state $|c\rangle_Y$ would hold even if the step 4 measurement of the Z register's state had been postponed to the present step, that is, even if the states of both registers had been simultaneously measured after performance of the quantum Fourier transform operation, as Shor[3] and Volovich[4] prescribe. For pedagogical purposes, however, it is preferable to measure the states of the two registers in two separate steps, as Williams and Clearwater[58] also recognize.

To grasp the implications of Eq. (45), it is helpful to consider first the exceptional case for which the order r is a power of 2. In this circumstance, Q exactly equals $2^y/r$ as explained in Sec. III C 4. Correspondingly, Eq. (45) becomes

$$P_c=(2^yQ)^{-1}\frac{\sin^2\pi c}{\sin^2(\pi rc/2^y)}. \tag{46}$$

Because c is an integer $0\leqslant c\leqslant 2^y-1$, Eq. (46) implies $P_c=0$ for any c other than values of c for which $rc/2^y$ is an integer d, as can occur because $2^y/r$ now is an integer. For such exceptional values of c, namely, the values of c for which

$$\frac{c}{2^y}-\frac{d}{r}=0, \tag{47}$$

the right side of Eq. (46) becomes 0/0, and we have to return to Eq. (43), where we see that except for the common factor $e^{2\pi ikc/2^y}$, every term in the sum over b for given c is unity. The number of terms in the sum is Q. So, when r is a power of 2 and the Y register is in the state described by the wave function Ψ_Y^{5S} of Eq. (43), the probability P_c that the Y register will be found in the basis state $|c\rangle_Y$ is zero except when c satisfies Eq. (47), in which case $P_c=(2^yQ)^{-1}Q^2=1/r$. Moreover, because $c<2^y$, the only values of c that can be observed are those corresponding to the integers d in the range $0\leqslant d<r$. Thus the total probability of observing these values of c is $rP_c=1$, as it must be.

Consider now the more general circumstance in which the order r is not purely a power of 2. Then there no longer exist values of c that satisfy Eq. (47) for every integer d such that $0\leqslant d<r$. In fact, if r is odd, we see that there are no values of c satisfying Eq. (47). Also we know from the discussion in Sec. III C 4 that $Q-2^y/r$ now equals a noninteger ξ, where $-1<\xi<1$. Accordingly, when r is not a power of 2, the numerator in Eq. (45) is not zero except possibly at a limited number of very special values of c. In other words, for most, perhaps all, values of c, P_c now is not zero. Nevertheless, for each integer d in the allowed range, the probability of observing the result c in a measurement on the Y register remains large for, and only for, these exceptional values of c that—though no longer satisfying Eq. (47)—come close to doing so. To quantify this assertion, note first that because $r<N/2$, the maximum allowed value of d/r [namely $1-1/r$)] surely is less than the maximum allowed value of $c/2^y$ (namely $1-1/2^y$, which is greater than $1-1/N^2$). Thus, because the spacing between successive values of $c/2^y$ is 2^{-y}, every allowed value of d/r either exactly satisfies Eq.

(47) for some value of c or else differs from some $c/2^y$ by no more than $2^{-y}/2$. In other words, when r is not purely a power of 2, Eq. (47) is replaced by

$$\left|\frac{c}{2^y}-\frac{d}{r}\right| \leqslant 2^{-y}/2, \tag{48}$$

with the assurance that for each allowed value of d/r, there exists a single $c=c_1$ satisfying Eq. (48) (except when the equality holds for such a c_1, in which case the equality holds for a second $c=c_2=c_1\pm 1$, corresponding to d/r lying exactly halfway between two successive values of $c/2^y$). Therefore, when c satisfies Eq. (48), we have

$$\frac{c}{2^y}=\frac{d}{r}+\varepsilon 2^{-y}, \tag{49}$$

where $-\frac{1}{2}\leqslant\varepsilon\leqslant\frac{1}{2}$.

If we employ Eq. (49) in Eq. (45), the probability of finding the Y register in this $|c\rangle_Y$ state (when r is no longer a power of 2) is seen to be

$$P_c=(2^yQ)^{-1}\frac{\sin^2(\pi r\varepsilon Q/2^y)}{\sin^2(\pi r\varepsilon/2^y)} \geqslant \frac{Q}{2^y}\frac{\sin^2(\pi r\varepsilon Q/2^y)}{(\pi r\varepsilon Q/2^y)^2}, \tag{50}$$

where we have used the fact that $\sin x\leqslant x$. The equality in Eq. (50) holds only when $\varepsilon=0$. Because 2^y is very large compared to both unity and $r<N/2$, the estimation of the right side of Eq. (50) by the replacement of Q by $2^y/r$ (Q actually differs from $2^y/r$ by a quantity ξ, $|\xi|<1$) can be seen to introduce an inconsequential error. In other words, the argument of the sine function on the right side of Eq. (50) may be taken to be $\pi\varepsilon$. Hence Eq. (50) yields

$$P_c\geqslant r^{-1}\frac{\sin^2\pi\varepsilon}{(\pi\varepsilon)^2}\geqslant r^{-1}\frac{1}{(\pi/2)^2}=\frac{4}{r\pi^2}, \tag{51}$$

where the second inequality results from recognizing that $\sin x/x$ is a decreasing function of x in the range $0\leqslant x\leqslant\pi$, and then replacing $|\varepsilon|$ by its maximum allowed value $\frac{1}{2}$. Because there is such a c and associated P_c for each of the r allowed values of d in Eq. (48), we conclude that even when r is not a power of 2, the total probability $P=rP_c$ of finding the Y register in a state $|c\rangle_Y$ for which c satisfies Eq. (48) is not less than $4/\pi^2\cong 0.4$.

This result for P has been obtained by Ekert and Josza;[5] it is larger than the value of P originally quoted by Shor.[3] It is clear from its derivation, however, that this lower bound of 0.4 (though rigorously derived) considerably underestimates the magnitude of P that is likely to be encountered in practice. For example, if in Eq. (51) $|\varepsilon|$ is replaced not by its maximum value but rather by its average value $\frac{1}{4}$, then Eq. (51) yields $P_c\geqslant 8/r\pi^2$, corresponding to $P\geqslant 0.81$. The use of the average value of $|\varepsilon|$ to estimate P is reasonable, because in general the value of ε depends on d, as can be seen from Eq. (49), remembering that $2^y/r$ is not an integer unless r is a power of 2.

7. Determine d/r. Attempt to infer the value of r

After a value of c has been obtained, that is, after the state measurement on register Y prescribed in step 6, it still is necessary to infer the value of r. To help understand how this inference is accomplished, I observe first that, because $r<N/2$, there can be only one permissible fraction d/r satisfying Eq. (48) for a given c. Here "permissible" means that d is an integer and $0<d<r<N/2$. To prove this assertion, note that if d_1/r_1 and d_2/r_2 are distinct permissible fractions, that is, if $d_1/r_1\neq d_2/r_2$, then

$$\left|\frac{d_1}{r_1}-\frac{d_2}{r_2}\right|=\left|\frac{d_1r_2-d_2r_1}{r_1r_2}\right|>\frac{1}{r_1r_2}>\frac{4}{N^2}, \tag{52}$$

because when $d_1/r_1\neq d_2/r_2$, the integer $(d_1r_2-d_2r_1)$ cannot equal 0. On the other hand, if d_1/r_1 and d_2/r_2 each satisfy Eq. (48) for the same c,

$$\begin{aligned}\left|\frac{d_1}{r_1}-\frac{d_2}{r_2}\right| &= \left|\left(\frac{c}{2^y}-\frac{d_2}{r_2}\right)-\left(\frac{c}{2^y}-\frac{d_1}{r_1}\right)\right| \\ &\leqslant \left|\frac{c}{2^y}-\frac{d_2}{r_2}\right|+\left|\frac{c}{2^y}-\frac{d_1}{r_1}\right| \\ &\leqslant 2(2^{-y}/2) \\ &= 2^{-y}<\frac{1}{N^2}. \end{aligned} \tag{53}$$

Because Eqs. (52) and (53) are inconsistent, the impossibility of finding two distinct d/r satisfying Eq. (48) for the same c is proved.

Suppose now that our state measurement on the Y register has yielded a $|c\rangle_Y$ state whose c satisfies Eq. (48). The actual evaluation of this d/r (now known to be unique) from Eq. (48) is performed by expanding $c/2^y$ into a continued fraction, as Shor[3] originally proposed. I shall not discuss here the construction of continued fraction expansions.[62] I provide an illustrative expansion in the following as well as an explanation of the relation between continued fraction expansions and gcd calculations (see Appendix D 2). Suffice it to say that the continued fraction expansion of any rational number x provides a series of fractions (with each fraction in lowest terms) called *convergents* to x, such that each successive convergent furnishes an improved approximation to x. A key theorem is that if a/b is a fraction satisfying[63]

$$\left|\frac{a}{b}-x\right|<\frac{1}{2b^2}, \tag{54}$$

then a/b is one of the continued fraction convergents to x. Equation (48) has the form of Eq. (54), with $x=c/2^y$ and $a/b=d/r$. Because $2^y>N^2$, the right side of Eq. (48) is less than $(2N^2)^{-1}$, which in turn is less than $(2r^2)^{-1}$ because $r<N/2$. This theorem implies that d/r must be one of the continued fraction convergents to $c/2^y$, that is, expanding $c/2^y$ in its series of continued fraction convergents inevitably will yield d/r in lowest terms. Note that this result demonstrates the critical importance of choosing the size y of the Y register to be much greater than N. Indeed, if $2^y<N^2/4$, the right side of Eq. (48) would be greater than $2N^2$, which would no longer ensure that d/r is one of the continued fraction convergents to $c/2^y$. Similarly, if $2^y<N^2/4$, the right side of Eq. (53) would be greater than $4/N^2$. Thus, Eq. (53) would no longer be inconsistent with Eq. (52), implying that

it is now no longer guaranteed that there is only one permissible d/r satisfying Eq. (48).

Let me illustrate this beautifully simple continued fraction method of determining d/r. To factor our illustrative $N=55$ via Shor's algorithm, a Y register of $y=12$ qubits will be employed, as shown in Sec. III C 1 ($2^{11}=2048<N^2=3025<2^y=4096$). The order of $n=37$ is $r=20$, the largest possible value of r for this N. For $d=11$, the value of d/r is exactly 0.55. We have $\frac{2252}{4096}=0.549\,80$, $\frac{2253}{4096}=0.550\,05$, and $2^{-y}/2=0.000\,12$, which is less than $0.000\,20=0.55-\frac{2252}{4096}$, but greater than $0.000\,05=\frac{2253}{4096}-0.55$. If we assume the state measurement on the Y register has yielded the state $|c\rangle_Y$ consistent with Eq. (48) for $r=20$ and $d=11$, the value of c must have been 2253. We now expand 2253/4096 in a continued fraction:

$$\frac{2253}{4096}=\frac{1}{4096/2253}=\frac{1}{1+1843/2253}$$

$$=\frac{1}{1+1/(2253/1843)}$$

$$=\frac{1}{1+1/(1+410/1843)} \tag{55a}$$

$$=\frac{1}{1+1/[1+1/(1843/410)]}$$

$$=\frac{1}{1+1/[1+1/(4+203/410)]}$$

$$=\frac{1}{1+1/\{1+1/[4+1/(2+4/203)]\}}, \tag{55b}$$

and so on. Dropping the fraction $\frac{410}{1843}$ in Eq. (55a) yields the first convergent, namely $\frac{1}{2}$; dropping the fraction $\frac{203}{410}$ in Eq. (55b) yields the second convergent, namely $\frac{5}{9}=0.5555$. Each of these two convergents differs from $\frac{2253}{4096}$ by an amount whose absolute value exceeds 0.000 12, that is, each of these convergents fails to satisfy Eq. (48) and so cannot equal the desired d/r. However, the third convergent, obtained by dropping the fraction $\frac{4}{203}$ in Eq. (55b), is $\frac{11}{20}$, confirming the theorem quoted in the preceding paragraph. Moreover, because we know $r<N/2$, which in this example is $\frac{55}{2}$, the result that $d/r=\frac{11}{20}$ immediately implies that $r=20$, because any other fraction equal to $\frac{11}{20}$, for example, $\frac{22}{40}$, inevitably has a denominator greater than $\frac{55}{2}$.

I next observe that because $r<N/2$, not merely less than N, it follows from Eq. (54) that even if the right side of Eq. (48) had been replaced by $2/N^2$, the values of $c/2^y$ satisfying the modified Eq. (48) would have continued fraction convergents equal to d/r. But $2^y>N^2$ implies $2/2^y<2/N^2$. In other words, if a state measurement on the Y register yields a $|c\rangle_Y$ whose c satisfies

$$\left|\frac{c}{2^y}-\frac{d}{r}\right|\leqslant 2(2^{-y}), \tag{56}$$

this $c/2^y$ also will have d/r as a continued fraction convergent of $c/2^y$, even though the value of c may not satisfy Eq. (48). Therefore, we have another reason (in addition to the desirability of using an average $|\varepsilon|$) for asserting that the quantity $4/\pi^2$ discussed following Eq. (51) greatly underestimates the probability of measuring states $|c\rangle_Y$ that can yield d/r.

To more accurately estimate this probability, note that if $c/2^y>d/r$ satisfies Eq. (48), then $c-2$, $c-1$, c, and $c+1$ will each satisfy Eq. (56). Similarly, if $c/2^y<d/r$ satisfies Eq. (48), then $c-1$, c, $c+1$, and $c+2$ will each satisfy Eq. (56). In either case, by adding the appropriate four P_c from Eq. (45), using $\sin x\leqslant x$ as in Eq. (50), and approximating Q by $2^y/r$ as was done in deriving Eq. (51), we find that the probability P_c' of measuring a state $|c\rangle_Y$ that will have d/r as a continued fraction convergent to $c/2^y$ is

$$P_c'\geqslant\frac{\sin^2\pi\varepsilon}{\pi^2 r}\left(\frac{1}{(1+\varepsilon)^2}+\frac{1}{\varepsilon^2}+\frac{1}{(1-\varepsilon)^2}+\frac{1}{(2-\varepsilon)^2}\right), \tag{57}$$

where $0\leqslant\varepsilon\leqslant\frac{1}{2}$, and the prime on P_c' indicates that we have summed over the appropriate four P_c. For $\varepsilon=\frac{1}{2}$, we obtain $P_c'=80/9\pi^2 r=0.90/r$; using the average $\varepsilon=\frac{1}{4}$ we obtain $P_c'=0.935/r$.

We return to our illustrative continued fraction expansion and readily verify that each of the continued fraction expansions of $\frac{2251}{4096}$, $\frac{2252}{4096}$, and $\frac{2254}{4096}$, as well as the Eq. (55) expansion of $\frac{2253}{4096}$, have $\frac{11}{20}$ as a convergent, consistent with our employment of Eq. (57) to estimate the probability of correctly inferring d/r from a state measurement $|c\rangle_Y$.

8. Repeat steps 2–7 until factorization of N is achieved

Inferring the value of r need not immediately lead to factorization of N, however. In the first place, as was mentioned in Sec. III A, the probability that r will meet the necessary requirements for being able to factor N, namely that r is even and satisfies Eq. (14), is only about $\frac{1}{2}$.[47] Thus although the probability of being able to infer d/r via a single measurement of the Y register is so high, namely over 90%, nevertheless, it will be necessary to run through the entire sequence of steps 2–7 at least twice on the average before a d/r whose r can be employed to factor N is obtained. The entire sequence must be repeated starting from step 2 (we don't have to make any new decisions about the sizes of the registers) because after step 7 the Y register is in whatever state $|c\rangle_Y$ was measured. The wave function of this state is nothing like the initial loading wave function Ψ_Y^{2S} of Eq. (31) from which the Shor algorithm operations departed, beginning with step 3. Also, unless we already have cleared register Z to the state $|0\rangle_Z$, the operations described in steps 2 and 3 will not yield the desired Ψ_C^{3S} of Eq. (37). Although the Y register wave function always will be brought to its initial loading form, Eq. (31), in these repetitions, that is, although step 2 always will be the same, the choice of n in step 3 had better be different. Otherwise, carrying the algorithm through to step 7 will again yield an r that cannot be employed to factor N.

Furthermore, even granting that the n selected in step 3 does possess an order r that is employable to factor N, inferring r from the computed d/r may not be as simple as the discussion in Sec. III C 7 suggests at first sight. Suppose that for our $N=55$, $n=37$, $r=20$ example, the register Y state measurement had yielded $c=2048$, which for $r=20$ satisfies

Eq. (48) with $d=10$. But the computer operator doesn't know $r=20$; all the operator knows is that $\frac{2048}{4096}=\frac{1}{2}$, the sole convergent (which has to be in lowest terms) to $\frac{2048}{4096}$. The operator immediately will discover $37^2 \equiv 49 \not\equiv 1 \pmod{55}$, so that 2 surely is not the order of 37, but then what? Each of the fractions $\frac{2}{4}, \frac{3}{6}, \frac{4}{8}, \ldots, \frac{13}{26}$, equals $\frac{1}{2}$ and has a denominator less than $\frac{55}{2}$, that is, each of these denominators could be the desired r. In principle, the operator could test the powers 37^2, 37^4, $37^6, \ldots \pmod{55}$ until he/she came to $37^{20} \equiv 1 \pmod{55}$. For the large N of interest, such as in RSA-309, persistently trying to determine r in this crude fashion after the register Y measurement has yielded a convergent with a denominator b for which $b \ll N$ and $n^b \not\equiv 1 \pmod{N}$ would be ridiculous and would negate the whole point of using Shor's algorithm. Shor[3] has suggested the operator should try a few small multiples of b, for example, $2b$ and $3b$; but after finding n^{2b} and $n^{3b} \not\equiv 1 \pmod{N}$, the operator seemingly would have little choice but to repeat steps 2–7 in the hope that for the newly selected n the newly measured $c/2^y$ would have a convergent whose denominator actually is r, not a factor of r.

How many times the operator may expect to have to repeat steps 2–7 before reliably inferring r (still assuming the operator has selected an n possessing an employable r) is difficult to say. A seeming overestimate of the expected number of such repetitions follows from considerations first advanced by Shor[3] and refined by Ekert and Josza.[5] The number of positive integers d less than r that are coprime to r is $\phi(r)$, where ϕ is Euler's totient function[64] (the subject of Appendix B). Then if P' is the probability (equal to at least 0.9 as we have seen) that a measurement on the Y register will yield a $c/2^y$ with a convergent equal to some d/r, $0 \leqslant d < r$, then $P'' = P' \phi(r)/r$ is the probability that the measurement will yield a convergent equal to a d/r where d is prime to r. For sufficiently large r, Ref. 5 quotes the theorem[65]

$$\frac{\phi(r)}{r} \geqslant \frac{0.56}{\ln \ln r} \cong \frac{1.17}{\log_2 \log_2 r}. \tag{58}$$

Because the typical r is expected to increase as N increases, Eq. (58) suggests that whatever may be the number of repetitions of steps 2–7 otherwise required (for example, repetitions because r is not always employable to factor N), these repetitions might need to be increased by about $\log_2 \log_2 r$ because of the just discussed complications associated with fractions d/r in Eq. (56) where d is not coprime to r.

This estimated increase in the required number of repetitions probably is an overestimate because it does not take into account the likelihood that the operator will infer r without recourse to repetitions when r is only a small multiple of the denominator b of the measured convergent, for example, when b equals $r/2$ or $r/3$. The operator also may be able to minimize the number of required repetitions by recognizing (as Shor[3] has remarked) that if starting with the same n yields two measured convergents which have denominators $b_1 \ll N/2$ and $b_2 \ll N/2$ with b_1 coprime to b_2, then the only way for r to be a multiple of b_1 and b_2 is for r to be a multiple of $b_1 b_2$, which now may be sufficiently large to ensure that r is either $2b_1b_2$ or $3b_1b_2$. For instance, for our $N=55$, $n=37$, $r=20$ example, if after obtaining the convergent $\frac{1}{2}$, the operator were to repeat steps 2–7 with the same n, and if this repeat should yield the convergent $\frac{3}{5}$, the operator would infer with high probability (greater than 0.9 as we discussed) that r is a multiple of 10. Once having discovered that $37^{10} \equiv 34 \pmod{55}$, the operator would infer with the same high probability that $r=20$, because $30 = 3 \times 10 > \frac{55}{2}$ and therefore cannot be r. Indeed, once having verified that $37^{20} \equiv 1 \pmod{55}$, the knowledge that $37^{10} \equiv 34 \pmod{55}$ immediately enables the factors 5 and 11 of $N=55$ to be determined.

IV. CONCLUDING REMARKS

We have now completed our presentation of the operations constituting Shor's algorithm. The algorithm involves the application of unitary operations at steps 2, 3, and 5. The estimated numbers of gates required to accomplish each of these steps are stated in the text under their respective headings. Let us denote these estimated numbers by ν_{q2}, ν_{q3}, and ν_{q5}, respectively. The estimated total number of gates required, denoted by ν_q in Eq. (28), equals $\nu_{q2}+\nu_{q3}+\nu_{q5}$. In the limit of very large N, the estimates ν_{q2} and ν_{q5} become negligible compared to ν_{q3}. Accordingly, Eq. (28) equates ν_q to ν_{q3}.[3,4] Equation (28) has not taken into account the operations, gate or otherwise, required to perform the state measurements postulated in steps 4 and 6. We have seen that the quantum computer can carry out the algorithm with no more than about $3L$ qubits. It is difficult to see why the required number ν_m of measurement operations, including the postmeasurement operations needed to restore the computer wave function to its starting form $\Psi_C^{(0)} = |0\rangle_Y |0\rangle_Z$, should be other than proportional to the number of qubits. Consequently, the failure to include state measurement operations in no way invalidates employing Eq. (28), which grows somewhat faster than L^2 with increasing N, to estimate the growth with N of the computing effort required to perform a factorization of N using Shor's algorithm.

If repetitions of the algorithm steps are necessary in practice, then these repetitions should be taken into account in any estimate, such as in Eq. (28), of the total number of gates required to determine a factorization of N [recall that Eq. (28) gives only the dependence of $\nu_q = \nu_{q3}$ on N]. If there are a number of repetitions that do not increase with N, for example, the expected number of repetitions associated with the fact that some r will not be employable to factor N, the inclusion of this number does not require a correction of Eq. (28). On the other hand, the number of repetitions suggested by Eq. (58), although very likely a considerable overestimate of the actual number of required repetitions associated with the desirability of measuring a d/r for which d is prime to r, probably does require some modification of Eq. (28). If we assume that the typical $r < N/2$ tends to be some fixed fraction of N, then for large N we can replace $\log_2 \log_2 r$ by $\log_2 \log_2 N$, thereby concluding that our earlier discussion of Eq. (28) should have been supplemented by an upper bound, $\nu_{\text{qub}}(N)$, on the expected number of universal quantum gates that will actually have to be employed in a Shor algorithm determination of the order r needed to factor N. The required modification is obtained via multiplication of $\nu_q(N)$ in Eq. (28) by $\log_2 L$, yielding

$$\nu_{\text{qub}}(N) = O[L^2 (\log_2 L)^2 (\log_2 \log_2 L)]. \tag{59}$$

Equation (59) only very minimally weakens our earlier conclusions from comparing Eqs. (7) and (28), or from computing the actual magnitude of $\nu_q(N)$ given by Eq. (29). For instance, whereas previously we concluded that a quantum computer that could factor RSA-309 in two weeks time should be able to factor RSA-617 in no more than about nine weeks, Eq. (59) leads to the conclusion that a quantum computer which can factor RSA-309 in no more than two weeks will factor RSA-617 in at most ten weeks.

After a c has been measured, as described in step 6, the following calculations still must be performed: (*i*) infer an r from the measured c, which generally involves a continued fraction expansion; (*ii*) verify that the inferred r satisfies Eqs. (10), (12), and (14) (as it must if this r is to be used to factor N), which involves computing n^r and $n^{r/2} \pmod N$; and (*iii*) obtain the factors p and q of N, which involves computing greatest common divisors. At present it is not contemplated that any of these calculations will be accomplished by a computer other than a purely classical one. The efforts required to accomplish these computations have not been included in Eq. (28), nor could they be, because Eq. (28) estimates the number of universal quantum gates required, not the number of classical computer bit operations as in Eq. (7). On the other hand, the efforts to perform these classical calculations are not irrelevant to any realistic estimate of the potential utility of Shor's algorithm for factoring increasingly large N. For none of these computations do the number of required bit operations increase with N more rapidly than the right side of Eq. (59) (see Appendix D). Consequently, Eq. (59) correctly exhibits the maximum expected growth with increasing $N=pq$ of the total computational effort, quantum plus classical, required to complete a factorization of N using Shor's algorithm. Correspondingly, the conclusions we have drawn from comparing Eqs. (7) and (28) remain valid, except for the very minimal weakening discussed immediately after Eq. (59), even though the derivation of Eq. (28) ignored the classical computer calculations presently inherent in the use of Shor's algorithm.

Until Shor produced his algorithm, it was generally believed that the computational effort required to factor $N=pq$ grew more rapidly with N than any polynomial in $L=\log_2 N$. Shor's demonstration that the use of a quantum computer could decrease this growth to slower than L^3 was astonishing, and has greatly accelerated attempts to construct a functioning quantum computer. The key Shor algorithm operation, the operation that enables the greatly diminished growth of the computational effort with N, is the quantum Fourier transform $\hat{U}_{FT}$ operation. The quantum Fourier transform is a direct generalization (to quantum mechanical basis states) of the classical discrete Fourier transform, which in turn is nothing more than a discretized version of the conventional Fourier integral transform. Thus it is not surprising that application of $\hat{U}_{FT}$ to the wave function Ψ_Y^{4S} of Eq. (39) yields a new wave function, namely Ψ_Y^{5S} of Eq. (44), where the probability P_c (that a measurement on the Y register will yield the basis state $|c\rangle_Y$) is large only for those values of c from which the periodicity with r inherent in Eq. (40) can be inferred. What is very remarkable, however, and what makes possible the comparatively slow increase with N of $\nu_q(N)$ in Eq. (28), is the fact that although the discrete Fourier transform calculation requires $O(NL)$ bit operations,[66] the operation $\hat{U}_{FT}$ can be accomplished with only $O(L^2)$ universal quantum gates, as stated following Eq. (43). It must be remembered that $\hat{U}_{FT}$ is a $2^y \times 2^y$ matrix, that is, at least an $N^2 \times N^2$ unitary matrix. Because an arbitrary unitary matrix of this dimensionality contains N^4 free parameters, one expects that reproducing a given $N^2 \times N^2$ unitary matrix will require a sequence of no fewer than $N^4/16$ one-qubit and two-qubit gates. This observation, based on trivial dimensional considerations, suggests that for most classical computer algorithms the growth of computational effort with number size will not be diminished merely by recasting the algorithm into a form usable in a quantum computer.

Finally, we remark that factorization of a number $N=pq$ via a quantum computer using Shor's algorithm actually has been accomplished.[67] Although the number factored, namely 15, is the smallest possible product of odd primes, the accomplishment is notable. It also is notable, however, that because $\phi(15)=4\times 2=8$, the only possible values of r were $r=2$ and $r=4$, meaning that in this quantum computer factoring demonstration, the value of r could be inferred from Eq. (47) for any chosen n coprime to and less than 15, without the complications attendant on the much more usual circumstance in which r has to be inferred from Eq. (48).

ACKNOWLEDGMENTS

I thank Paul Reilly for numerous enlightening discussions, especially on the RSA system. I am indebted to Joseph Burdis for carefully checking the manuscript, including its references. I also am indebted to Sam Scheinman for data on RSA enciphering and deciphering.

APPENDIX A: CONGRUENCE MANIPULATIONS, ILLUSTRATIVE RSA OPERATIONS, AND PERIODICITY OF REMAINDERS f_j

A comparison of Eqs. (1) and (5) illustrates the proposition that (subject to the important proviso that all the congruences must have the same modulus m) congruences like Eqs. (1)–(5) have the useful property that in many respects they can be manipulated as if they were equalities, i.e., as if the congruence symbol $\equiv$ were the equality symbol $=$. For example Eq. (1) and

$$x \equiv y \pmod m \tag{A1}$$

imply

$$ax \equiv by \pmod m \tag{A2}$$

and

$$bx \equiv ay \pmod m. \tag{A3}$$

Accordingly Eq. (1) implies

$$a^z \equiv b^z \pmod m \tag{A4}$$

for any positive integer z. Equations (A2)–(A4) can be trivially demonstrated[28] remembering that Eq. (1) means $a=b+wm$ for some positive or negative integer w. There are a few permissible manipulations of equalities that have no congruence counterparts, but any such manipulations are not relevant here. Unless explanatory comments seem required, therefore, the remainder of this appendix will manipulate congruences as if they were equalities.

The use of congruence manipulations to conveniently compute Alice's $S=21$, 14, 51, 1, 13, 27, 10, 1, 9, 8, 49, 51 from her illustrative $C=21$, 9, 6, 1, 7, 3, 10, 1, 4, 2, 14, 6 will now be shown. Our illustrative RSA key number and encryption exponent, to be inserted into Eq. (2) along with each c in C, are $N=55$ and $e=23$, respectively. Consider Alice's first $c=21$. Instead of determining the corresponding s by tediously computing $c^e=(21)^{23}$ and then dividing by 55, Alice proceeds as follows:

$$(21)^2=441\equiv 1 \pmod{55}, \tag{A5}$$

$$(21)^{22}\equiv(1)^{11}\equiv 1 \pmod{55}, \tag{A6}$$

$$(21)^{23}=(21)(21)^{22}\equiv 21(1)\equiv 21 \pmod{55}. \tag{A7}$$

So the first s in S turns out to equal the first $c=21$. The second s is obtained not quite as simply, but much more easily than having to exactly compute 9^{23}, namely,

$$(9)^2=81\equiv 26 \pmod{55}, \tag{A8}$$

$$(9)^4\equiv(26)^2=676\equiv 16 \pmod{55}, \tag{A9}$$

$$(9)^8\equiv(16)^2=256\equiv 36\equiv -19 \pmod{55}, \tag{A10}$$

$$(9)^{10}\equiv(-19)(26)=-494\equiv -54\equiv 1 \pmod{55}, \tag{A11}$$

$$(9)^{20}\equiv(1)^2\equiv 1 \pmod{55}, \tag{A12}$$

$$(9)^{23}=(9)^{20}(9)^2(9)\equiv(26)(9)=234\equiv 14 \pmod{55}. \tag{A13}$$

Thus the second s is 14. Similar congruence manipulations on C readily yield the complete S. Similarly, it is readily verified that Bob's secret decryption exponent $d=7$ really does decipher this S into C, namely, that in accordance with Eq. (3), $(21)^7=(21)(21)^6\equiv 21 \pmod{55}$ using Eq. (A5); $(14)^2=196\equiv 31$, $(14)^4\equiv 961\equiv 26$, $(14)^7\equiv(14)(31)(26)\equiv(14)(36)\equiv 9 \pmod{55},\ldots$.

The permissibility of these congruence manipulations also immediately implies the periodicity of the remainders f_j defined by Eq. (8). Using Eq. (10), we see that $f_{j+r}\equiv n^{j+r}=n^j n^r\equiv n^j\equiv f_j \pmod{pq}$, implying that $f_{j+r}=f_j$, because by definition all the f_j are positive numbers less than N; similarly $f_{j+2r}\equiv n^{j+r}n^r\equiv n^{j+r}\equiv f_j \pmod{pq}$. It also is readily seen that all the f_j, $1\leqslant j\leqslant r$, are different. For suppose $f_a=f_b$, where each of $a\neq b$ lies in the range of j. Suppose further that $a<b$. Then $n^b\equiv n^a \pmod{pq}$, implying $n^b-n^a=n^a(n^{b-a}-1)$ is divisible by pq. This means $n^{b-a}-1$ must be divisible by pq, because n has been chosen to be coprime to pq. On the other hand, it is not possible to have $n^{b-a}\equiv 1 \pmod{pq}$ because by definition r is the smallest value of j for which $n^j\equiv 1 \pmod{pq}$.

APPENDIX B: EULER'S TOTIENT FUNCTION AND THE PROOF $r<N/2$ FOR $N=pq$

For any positive integer m, Euler's totient function[64] $\phi(m)$ is the number of positive integers less than m that are coprime to m, where by definition $\phi(m)$ always is less than m. Euler proved[68] that if a is coprime to m, then

$$a^{\phi(m)}\equiv 1 \pmod{m}. \tag{B1}$$

Let us calculate $\phi(N)$ for $N=pq$, where p and q are odd primes. The only numbers less than N that are not coprime to N are multiples of p and q. There are $q-1$ integers p, $2p,\ldots,(q-1)p$ less than N; similarly there are $p-1$ multiples of q that are less than N. Because none of these numbers can coincide and be less than N,

$$\phi(N)=N-1-[(p-1)+(q-1)]=pq-p-q+1$$
$$=(p-1)(q-1). \tag{B2}$$

Evidently the RSA ϕ introduced in Sec. II A and employed in Eqs. (4) and (9) is $\phi(pq)$. The explicit dependence on $N=pq$ was dropped in those equations because no confusion could result. Equally evident is that Eq. (B1) immediately implies Eq. (9). I note that if n is not coprime to pq, that is, if n and pq have a common factor $x>1$, then f_j in Eq. (8) also must be divisible by x, as is immediately seen because Eq. (8) means $n^j-f_j=ypq$, for y equal to an integer. Consequently, Eq. (10) cannot hold for any integer r unless n actually is coprime to pq.

If n is coprime to pq, Eq. (9) is supplemented by

$$n^{(p-1)(q-1)/2}\equiv 1 \pmod{pq}, \tag{B3}$$

where n is coprime to pq. To prove Eq. (B3) it is convenient to start from the form taken by Eq. (B1) when m is an odd prime p. Evidently $\phi(p)=p-1$, so that

$$a^{p-1}\equiv 1 \pmod{p}, \tag{B4}$$

where a is coprime to p. Equation (B4) is known as Fermat's Little Theorem stated by Fermat in 1640.[69] Because q also is an odd prime, $(q-1)/2$ is an integer, so that Eq. (B4) implies, using Eq. (A4),

$$a^{(p-1)(q-1)/2}\equiv 1 \pmod{p}. \tag{B5}$$

But if a also is coprime to q, then it similarly is true that

$$a^{(q-1)(p-1)/2}\equiv 1 \pmod{q}. \tag{B6}$$

If a is coprime to both p and q, however, then a is coprime to pq, that is, a is an n as defined at the outset of Sec. III A. Furthermore, for any positive integer z, if $z-1$ is separately divisible by a prime p and by another prime q, then $z-1$ is divisible by the product pq. Hence Eqs. (B5) and (B6) imply Eq. (B3). Equation (B3) in turn implies that the order r of any n modulo $N=pq$ is less than or equal to $\phi(N)/2$, so that $r<N/2$, an inequality that is crucial to the derivation of the important Eq. (57).

The probability that a randomly selected positive integer less than N will be coprime to N is

$$\frac{\phi(N)}{N-1}=1-\frac{p-1+q-1}{N-1}, \tag{B7}$$

using Eq. (B2). For actual RSA key numbers, for example, RSA-309, the right side of Eq. (B7) will be indistinguishable from unity for all practical purposes. For instance, if the smaller of p and q is not less than $N^{1/4}$, the larger of p and q will be no greater than $N^{3/4}$, and the right side of Eq. (B7) is approximately $1-N^{-1/4}$, which, even for a key number as small as RSA-155, differs from unity by approximately 10^{-39}. Correspondingly, for actual RSA key numbers the magnitude of $\phi(N)$ can be taken equal to N for all practical purposes. It is worth noting that because $\phi(N)$ also equals the number of integers i coprime to N in the ranges $N+1<i<2N$, $2N+1<i<3N$, etc., the probability that a randomly selected integer less than N^2 (or less than 2^y) will be coprime to N also can be taken equal to unity for all practical purposes.

APPENDIX C: PROOF THAT THE RSA SYSTEM CORRECTLY DECIPHERS

Equations (2) and (3), together with the definition of N, imply

$$u \equiv (c^e)^d = c^{ed} \pmod{pq}. \tag{C1}$$

Because d and e are positive integers by definition, Eq. (4) implies

$$de = 1 + z\phi = 1 + z(p-1)(q-1), \tag{C2}$$

where z is a positive integer. Because by definition u and c are positive integers less than $N=pq$, knowing $u \equiv c \pmod{pq}$ implies $u=c$. Thus to prove that the RSA system enables Bob to correctly decipher Alice's message, I need to show that

$$c^{1+z(p-1)(q-1)} \equiv c \pmod{pq}. \tag{C3}$$

If c is coprime to N, then Eq. (9) implies

$$c^{z(p-1)(q-1)} \equiv 1 \pmod{pq}, \tag{C4}$$

from which Eq. (C3) immediately follows after multiplying both sides of Eq. (C3) by c. If c is not coprime to pq, $c < N$ is divisible by p or q but not by both. Suppose that c is divisible by q, that is, suppose $c=bq$, with b a positive integer less than p. Then Eq. (B4) holds for $a=c$ and implies

$$c^{z(p-1)(q-1)} \equiv 1 \pmod{p}. \tag{C5}$$

But if $x-y$ is divisible by p, then $q(x-y)$ is divisible by qp, implying further that $bq(x-y)$ is divisible by qp. Hence Eq. (C5) implies

$$bqc^{z(q-1)(p-1)} \equiv bq \pmod{qp}. \tag{C6}$$

Equation (C6) is Eq. (C3) for the case that $c=bq$. Correspondingly, Eq. (C3) will hold if c is divisible by p. We conclude that Eq. (C3) holds for every c in Alice's cryptogram whether c is coprime to N or not. This completes the demonstration that the RSA system enables Bob to correctly decipher Alice's cryptogram.

APPENDIX D: CLASSICAL COMPUTER CALCULATIONS RELEVANT TO SHOR'S ALGORITHM FACTORIZATION

This appendix discusses the various classical computer calculations discussed in this paper. The results 1–3 in the following are the bases for the assertions made in Sec. IV about the growth with N of the classical computer calculations required to factor $N=pq$ using Shor's algorithm. How Bob can determine his decryption exponent d from $\phi = \phi(N)$ and his encryption exponent e is described in Appendix D 4.

1. Greatest common divisors and the Euclidean algorithm

A convenient method for computing the gcd of two positive integers was first described by Euclid. My discussion of the Euclidean algorithm closely follows Rosen.[70] Suppose the integer $x \geq 1$ is the gcd of the two positive integers s_0 and s_1, where $s_0 > s_1 > 1$. The Euclidean algorithm determines x as follows. Divide s_0 by s_1, thereby obtaining the remainder $s_2 \geq 0$. By definition $s_2 < s_1$ and

$$s_0 = z_0 s_1 + s_2, \tag{D1}$$

where z_0 is a non-negative integer and $0 \leq s_2 < s_1$. Because x is a divisor of s_0 and s_1, Eq. (D1) implies x is a divisor of s_2. If we proceed in this fashion, always dividing the previous divisor s_j by the previous remainder s_{j+1}, we obtain a sequence of remainders $s_2, s_3, \ldots, s_{j+1}, s_{j+2}, \ldots$, each of which is a multiple of x. Moreover, because each s_j is greater than s_{j+1}, the sequence eventually must terminate with some $s_{k+2}=0$, that is, eventually there will be the simple equation

$$s_k = z_k s_{k+1}. \tag{D2}$$

It now can be seen that s_{k+1} is not merely a multiple of x (which is the gcd of s_0 and s_1), but rather $s_{k+1}=x$. Equation (D2) shows that s_k is a multiple of s_{k+1}. The preceding equation in the series, namely

$$s_{k-1} = z_{k-1} s_k + s_{k+1}, \tag{D3}$$

then implies s_{k-1} also is a multiple of s_{k+1}. Thus, proceeding back through the series of equations that led from Eq. (D1) to Eq. (D3), it can be concluded that both s_0 and s_1 are multiples of s_{k+1}. Hence, s_{k+1} must be a divisor of x, the gcd of s_0 and s_1. But we already have shown that x is a divisor of s_{k+1}. Consequently, x must be identical to s_{k+1}, the last remainder before $s_{k+2}=0$. If $s_{k+1}=1$, then s_1 is coprime to s_0.

I will illustrate the use of the Euclidean algorithm to find the factor 5 of $N=55$ when, as explained at the end of Sec. III A, it is deduced for $n=12$ that $f_2+1=35$ must be divisible by one of the factors of 55. We have $55=1\times35+20$; $35=1\times20+15$; $20=1\times15+5$; $15=3\times5+0$. Therefore 5 is the gcd of 55 and 35. Similarly, suppose we had decided to verify that 12 actually is coprime to 55. Now we have $55=4\times12+7$; $12=1\times7+5$; $7=1\times5+2$; $5=2\times2+1$; $2=2\times1+0$. So 1 is the gcd of 12 and 55, that is, 12 really is coprime to 55.

How many classical computer bit operations are required to obtain the gcd of two large numbers N_1 and $N_2 < N_1$ via the Euclidean algorithm? Define $L_1 = \log_2 N_1$, and $L_2 = \log_2 N_2$. As discussed following Eq. (29), for large N_1, N_2 the quantities L_1, L_2 differ negligibly from the number of digits in the binary expansions of N_1, N_2, respectively. Then according to a theorem by Lamé,[70] the number of divisions needed to find the gcd of N_1 and N_2 using the Euclidean algorithm is at most $O(L_1)$. The number of bit operations in any one of these divisions hardly can exceed the number of bit operations in the first of those divisions, where the dividend N_1 and divisor N_2 are at their respective maximum values. Although at first sight the number of bit operations required to divide N_1 by N_2 is $O(L_1 L_2)$,[71] in actuality, there exist[72] sophisticated classical computer algorithms which for large N_1, N_2 reduce the number of bit operations required for this division to $O[L_1(\log_2 L_1)(\log_2 \log_2 L_1)]$. Consequently, the number of computer bit operations required to obtain the gcd of two large numbers N_1 and N_2 using the Euclidean algorithm is $O[L_1^2(\log_2 L_1)(\log_2 \log_2 L_1)]$.

Returning to the discussion in Sec. IV of the classical computer calculations required for factoring using Shor's algorithm, the two numbers whose gcd is required always will be no larger than $N=pq$ and $1+f_{r/2}$, where according to Eq. (8) every f_j is less than N by definition. Once an r permitting factorization of N has been inferred, only a single gcd computation will be needed to complete the factorization of N. No such gcd calculation is needed until a usable r has been inferred. It follows that the number of bit operations required for the gcd calculations involved in factoring $N = pq$ using Shor's algorithm will not grow faster than $O[L^2(\log_2 L)(\log_2 \log_2 L)]$, the same growth rate as given by Eq. (28).

2. Continued fraction expansions

Rosen[62] has explicitly demonstrated that the divisions performed in finding the gcd of the positive integers s_0 and s_1 via the Euclidean algorithm are the same as the divisions performed in constructing the continued fraction expansion of the fraction s_1/s_0. By way of illustration, suppose we seek the gcd of the integers 2253 and 4096 whose ratio was expanded in the continued fraction of Eq. (55). We have $4096=1\times2253+1843$; $2253=1\times1843+410$; $1843=4\times410+203$; $410=2\times203+4$; and so on. Evidently the divisions performed to obtain these relations are identical with those performed in constructing the right side of Eq. (55). Thus to estimate the number of bit operations required to compute the continued fraction convergents of any one $c/2^y$ measured as described in Sec. III C 6, the result obtained in Appendix D 1 is immediately applicable. It is necessary only to observe that for sufficiently large N, the value of $y=\log_2 2^y$ differs negligibly from $2L=\log_2 N^2$. Accordingly, the number of bit operations required to perform a typical continued fraction expansion of a measured $c/2^y$ should be $O[(2L)^2(\log_2 2L)\times(\log_2\log_2 2L)]=O[L^2(\log_2 L)(\log_2\log_2 L)]$, the same result as obtained in Appendix D 1 for the Shor algorithm gcd calculation.

Unlike the gcd case, however, a continued fraction expansion is required every time a $c/2^y$ is measured. The expected number of repetitions of such measurements has been discussed in Sec. III C 8 and in Sec. IV. Those discussions indicated that a probable overestimate of the required number of repetitions is $2 \log_2 L$, implying that the overall number of bit operations required to perform the continued fraction expansions during factoring by Shor's algorithm may grow with increasing N as fast as, but no faster than, the right side of Eq. (59).

3. Modular exponentiation

Verifying that $n^r \equiv 1 \pmod N$, and then computing $n^{r/2} \equiv f_{r/2} \pmod N$, so as hopefully to factor $N=pq$ via Eq. (11), involves *modular exponentiation*. Volovich[4] has sketched the proof that the number of bit operations required to calculate $n^j \pmod N$ on a classical computer is $O[L^2(\log_2 L)(\log_2\log_2 L)]$, that is, grows with N as does the right side of Eq. (28). A few repetitions of these exponentiations may be necessary because the probability that a chosen n will yield an r permitting factorization of N via Eq. (11) is only about $\frac{1}{2}$. A few more exponentiations may be necessary to rule out as possible values of r the denominators b (and small multiples thereof) of convergents a/b to measured $c/2^y$ when $a/b=d/r$ with $b \ll r$. It does not appear, however, that as many repeated exponentiations ever will be required as the $O(\log_2 L)$ repetition factor inferred from Eq. (58). Consequently, the number of bit operations needed to perform the classical computer modular exponentiations that arise during factorization via Shor's algorithm may grow with increasing N as fast as, but surely no faster than, the right side of Eq. (59).

4. Finding the decryption exponent

We need to solve Eq. (4) for d, knowing e and ϕ. There is a known closed formula for $\phi(\phi)$, the totient function of ϕ, in terms of the prime factors of ϕ.[73] Thus if we could factor ϕ, we immediately could find d. Namely because e is coprime to ϕ (recalling Sec. II A), using Eq. (B1) implies

$$e^{\phi(\phi)} \equiv 1 \pmod{\phi}. \tag{D4}$$

Consequently Eq. (4) is solved by

$$d \equiv e^{\phi(\phi)-1} \pmod{\phi}. \tag{D5}$$

When N is of the magnitude of modern RSA key numbers, factoring a $\phi=(p-1)(q-1)\cong N$ can be difficult, though perhaps not as difficult as factoring $N=pq$ itself. In practice, therefore, d probably would be determined as follows. Equation (4) means there is an integer k such that

$$ed = 1 + k\phi. \tag{D6}$$

Equation (D6) is a *Diophantine equation* in the unknowns k and d, whose solution can be found[74] by working backward from the set of equations constituting the Euclidean algorithm for the gcd of e and ϕ.

I will illustrate this method of solving Eq. (4) for $N=55$. We have $\phi=40$, and have chosen $e=23$. The Euclidean algorithm equations for obtaining the gcd of 40 and 23 are $40=1\times23+17$; $23=1\times17+6$; $17=2\times6+5$; $6=1\times5+1$; $5=5\times1+0$, verifying that e is coprime to N. Now, working backward we have $6-5=1$; $5=17-2\times6$, so $6-(17-2\times6)=3\times6-17=1$; $6=23-17$, so $3\times(23-17)-17=3\times23-4\times17=1$; $17=40-23$, so $3\times23-4\times(40-23)=7\times23-4\times40=1$. This last equation is of the form of Eq. (D6), and implies $7\times23\equiv1$ (mod 40). Therefore the desired d equals 7, as asserted at the end of Sec. II A.

It is apparent that the computing effort required of Bob to determine his d via this procedure is utterly negligible compared to the computing effort he will endure in decrypting the many messages he expects to receive from Alice.

[1] N. D. Mermin, "From cbits to qbits: Teaching computer scientists quantum mechanics," Am. J. Phys. **71**, 23–30 (2003).

[2] L. K. Grover, "From Schrodinger's equation to the quantum search algorithm," Am. J. Phys. **69**, 769–777 (2001).

[3] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proceedings of the 35th Annual Symposium on the Foundations of Computer Science*, edited by S. Goldwasser (IEEE Computer Society, Los Alamitos, CA, 1994), pp. 124–134. P. W. Shor, SIAM J. Comput. **26**, 1484–1509 (1997) provides an expanded version of Shor's original paper.

[4] I. V. Volovich, "Quantum computing and Shor's factoring algorithm," quant-ph/0109004.

[5] A. Ekert and R. Josza, "Quantum computation and Shor's factoring algorithm," Rev. Mod. Phys. **68**, 733–753 (1996).

[6] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge U.P., Cambridge, 2000), pp. 232–247.

[7] C. P. Williams and S. H. Clearwater, *Explorations in Quantum Computing* (Springer, New York, 1998), pp. 130–145.

[8] G. Johnson, *A Shortcut Through Time* (Knopf, New York, 2003), pp. 66–82.

[9] J. Brown, *The Quest for the Quantum Computer* (Simon and Schuster, New York, 2000), pp. 170–188.

[10] See, for example, ⟨arXiv.org/archive/quant-ph⟩. See also ⟨www.eg.bucknell.edu/ ~ dcollins/research/qcliterature.html⟩.

[11]An exposition (suitable for the nonspecialist readers of this journal) of what is now termed the RSA public key system can be found in Ref. 7, pp. 122–127.

[12]An important reference, probably useful to cryptography specialists only, is A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography* (CRC, Boca Raton, FL, 1996), Chap. 8.

[13]The RSA system was first proposed by R. Rivest, A. Shamir, and L. Adleman, ''On digital signatures and public-key cryptosystems,'' MIT Laboratory for Computer Science Technical Report MIT/LCS/TR-212 (January 1979).

[14]A. Ekert, ''From quantum code-making to quantum code-breaking,'' quant-ph/9703035.

[15]D. Kahn, *The Codebreakers: The Story of Secret Writing* (Scribner, New York, 1996). For the definitions adopted here, see pp. xv–xviii and 989.

[16]S. Singh, *The Code Book* (Doubleday, New York, 1999), especially Chaps. 6 and 7.

[17]N. Gisin, Grégoire Ribordy, Wolfgang Tittel, and Hugo Zbinden, ''Quantum cryptography,'' Rev. Mod. Phys. **74**, 145–195 (2002), pp. 147–148, and Ref. 16, pp. 268–273.

[18]Reference 15, pp. 71–88.

[19]Reference 15, pp. 93–105.

[20]''The Gold-Bug,'' published in 1843. See, for example, *Complete Stories and Poems of Edgar Allan Poe* (Doubleday, New York, 1966), pp. 70 and 819.

[21]''The Adventure of the Dancing Men.'' See, for example, *The Complete Sherlock Holmes* (Doubleday, New York, 1966), p. 593.

[22]Reference 16, pp. 20–25, and Ref. 15, pp. 99–105, provide detailed illustrative cryptanalyses of such cryptograms.

[23]Reference 16, Chap. 4, describes the Enigma machine and recounts the remarkable story of how its cryptograms were cryptanalyzed. See also A. Hodges, *Alan Turing: The Enigma* (Simon and Schuster, New York, 1983), Chap. 4.

[24]Actually it is possible, though intrinsically inconvenient, for Alice and Bob to establish a secure key via conventional communication channels without meeting, as was discovered in 1976; see Ref. 16, pp. 253–267. Secure key distribution also is possible (in theory at least) via ''quantum channels,'' for example, channels that carry pairs of spin $\frac{1}{2}$ particles whose spin orientations can be measured by Alice and Bob; see Ref. 14. These secure key distribution schemes are beyond the scope of this paper.

[25]I do not pretend that this analogy between cryptographic keys and safes is original. See, for example, Ref. 17.

[26]Reference 16, pp. 245–249 and 379.

[27]ASCII is the acronym for the American Standard Code for Information Interchange. For more information on ASCII, see ⟨www.jimprice.com/jim-asc.htm⟩, especially the link to a decimal-to-ASCII chart.

[28]See any textbook on elementary number theory, for example, K. H. Rosen, *Elementary Number Theory and its Applications* (Addison-Wesley, Reading, MA, 1993), pp. 119–125.

[29]Reference 12, especially p. 292.

[30]''How large a key should be used in the RSA cryptosystem?'' ⟨www.rsasecurity.com/rsalabs/node.asp?id=2218⟩. See also ''TWIRL and RSA key size,'' ⟨www.rsasecurity.com/rsalabs/node.asp?id=2004⟩.

[31]A. Ekert, ''Quantum cryptoanalysis—Introduction'' (as updated by Wim van Dam, June 1999), ⟨www.qubit.org/library/intros/cryptana.html⟩.

[32]R. Roskies, Scientific Director Pittsburgh Supercomputing Center, private communication.

[33]''How Old is the Universe?'' (NASA 4/30/04) at ⟨map.gsfc.nasa.gov/m_uni/uni_101age.html⟩.

[34]Reference 9, pp. 164–166.

[35]R. Crandall and C. Pomerance, *Prime Numbers. A Computational Perspective* (Springer, New York, 2001), pp. 225–232.

[36]''What is the RSA Factoring Challenge?'' ⟨www.rsasecurity.com/rsalabs/node.asp?id=2192⟩.

[37]Reference 35, pp. 242–258.

[38]''What are the best factoring methods in use today?'' ⟨www.rsasecurity.com/rsalabs/node.asp?id=2190⟩.

[39]''RSA-160 is factored!'' ⟨www.rsasecurity.com/rsalabs/node.asp?id=2097⟩.

[40]''The RSA challenge numbers,'' ⟨www.rsasecurity.com/rsalabs/node.asp?id=2093⟩.

[41]Reference 35, p. 265.

[42]N. Koblitz, *A Course in Number Theory and Cryptography* (Springer, New York, 1987), pp. 3–4.

[43]Reference 7, p. 35.

[44]P. Ribenboim, *The New Book of Prime Number Records* (Springer, New York, 1995), p. 156, writes: ''It is fairly easy, in practice, to produce large primes. It is, however, very difficult to produce a theoretical justification for the success of the method.'' See also P. Ribenboim, ''Selling Primes,'' Math. Mag. **68**, 175–182 (1995). The essential point is that finding the primes p and q which will be multiplied to construct N can be accomplished in computing times at most polynomial in $L=\log_2 N$, whereas factoring N to find its prime factors p and q requires computing times subexponential in L (as we have discussed, assuming only classical computers are available).

[45]A recent test run demonstrated that even with an RSA key number of 2048 binary bits (that is, an RSA-617) a message consisting of approximately 32 000 ASCII characters could be routinely enciphered and deciphered in times of the order of seconds and at most minutes, respectively, employing merely a 700 MHz desktop computer (hardly a supercomputer). For example, using block sizes of 52 ASCII characters (recall Sec. II C), the encryption and decryption times were 1.46 and 30.3 s, respectively. Sam Scheinman, software engineer consultant, private communication.

[46]Reference 28, pp. 278–279.

[47]Reference 35, p. 386.

[48]Reference 42, p. 94.

[49]A. Odlyzko, ''Discrete logarithms: The past and the future,'' Designs, Codes, Cryptogr. **19**, 129–145 (2000).

[50]J. Eisen and M. Wolf, ''Quantum computing,'' quant-ph/0401019, cf, especially, p. 16.

[51]See, for example, Ref. 6, Chap. 7.

[52]D. J. Griffiths, *Introduction to Quantum Mechanics* (Prentice Hall, Englewood Cliffs, NJ, 1994), pp. 154–159.

[53]V. Scarani, ''Quantum computing,'' Am. J. Phys. **66**, 956–960 (1998).

[54]Reference 52, p. 12.

[55]Reference 1, especially Eq. (35).

[56]See, for example, Ref. 6, Chap. 4.

[57]Reference 35, p. 7.

[58]Reference 7, especially pp. 136–137.

[59]See, for example, Ref. 6, pp. 18–19.

[60]Reference 6, pp. 194–198.

[61]Reference 6, pp. 217–220.

[62]Reference 28, pp. 394–403.

[63]See, for example, Ref. 5. These authors refer to G. H. Hardy and E. M. Wright, *An Introduction to the Theory of Numbers* (Clarendon, Oxford, 1965), Sec. 10.15, for a proof of the theorem.

[64]Reference 35, p. 11.

[65]Reference 44, pp. 319–320.

[66]D. E. Knuth, *The Art of Computer Programming* (Addison-Wesley, Reading, MA, 1981), 3rd ed., Vol. 2, pp. 290 and 300 (see Problem 8).

[67]L. Vandersypen, Matthias Steffen, Gregory Breyta, Costantino S. Yannoni, Mark H. Sherwood, and Isaac L. Chuang, ''Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance,'' Nature (London) **414**, 883–887 (2001).

[68]Reference 28, pp. 201–204.

[69]Reference 28, p. 187; L. E. Dickson, *Modern Elementary Theory of Numbers* (University of Chicago, Chicago, 1939), p. 12, quotes the date of Fermat's Little Theorem.

[70]Reference 28, pp. 80–84.

[71]Reference 42, pp. 6–7.

[72]Reference 28, p. 61; Ref. 66, p. 295.

[73]Reference 28, p. 210.

[74]Reference 28, pp. 132–133.