Quantum algorithms for algebraic problems

Andrew M. Childs*

Department of Combinatorics and Optimization and Institute for Quantum Computing, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1

Wim van Dam†

Departments of Computer Science and Physics, University of California, Santa Barbara, California 93106, USA

(Published 15 January 2010)

Quantum computers can execute algorithms that dramatically outperform classical computation. As the best-known example, Shor discovered an efficient quantum algorithm for factoring integers, whereas factoring appears to be difficult for classical computers. Understanding what other computational problems can be solved significantly faster using quantum algorithms is one of the major challenges in the theory of quantum computation, and such algorithms motivate the formidable task of building a large-scale quantum computer. This article reviews the current state of quantum algorithms, focusing on algorithms with superpolynomial speedup over classical computation and, in particular, on problems with an algebraic flavor.

DOI: 10.1103/RevModPhys.82.1 PACS number(s): 03.67.Lx

CONTENTS

*amchilds@uwaterloo.ca
†vandam@cs.ucsb.edu

I. INTRODUCTION

In the early 1980s, Manin (1980) and Feynman (1982) independently observed that computers built from quantum mechanical components would be ideally suited to simulating quantum mechanics. Whereas brute-force classical simulation of a system of n quantum particles (say, two-level atoms) requires storing 2^n complex amplitudes and hence exponentially many bits of information, a quantum computer can naturally represent those amplitudes using only n quantum bits. Thus, it is natural to expect a quantum mechanical computer to outperform a classical one at quantum simulation.[1]

The perspective of quantum systems as abstract information processing devices subsequently led to the identification of concrete tasks, apparently unrelated to quantum mechanics, for which quantum computers have a quantifiable advantage. Deutsch (1985) gave the first such example, a black-box problem that requires two queries to solve on a classical computer but that can be solved with only one quantum query. A series of related results (Deutsch and Jozsa, 1992; Bernstein and Vazirani, 1997) gave increasingly dramatic separations between classical and quantum query complexities, culminating in an example from Simon (1997) providing an exponential separation. Building on this work, Shor (1997) discovered in 1994 that a quantum computer could efficiently factor integers and calculate discrete logarithms. Shor's result drew considerable attention to the concept of quantum information processing [see Ekert and Jozsa (1996) for an early review], and since then, the design and analysis of quantum algorithms has become a vibrant research area.

Quantum computers achieve speedup over classical computation by taking advantage of interference between quantum amplitudes. Of course, interference occurs in classical wave mechanics as well, but quantum mechanics is distinguished by the ability to efficiently represent a large number of amplitudes with only a few quantum bits.[2] In Shor's algorithm and its predecessors, the "exponential interference" leading to quantum speedup is orchestrated using a unitary operation called the *quantum Fourier transform* (QFT), an algebraic operation. In this article, we review the state of the art in quantum algorithms for *algebraic problems*, which can be viewed as continuations of the line of work leading from Deutsch to Shor. Many, though not all, of these algorithms make use of the QFT in some capacity.

Before beginning our exploration of quantum algorithms for algebraic problems, we summarize the development of quantum algorithms more generally. It has sometimes been said that there are really only two quantum algorithms: Shor's and Grover's. We hope that this article will, in some small way, help to dispel this pernicious myth. While it is difficult to compete with the impact of Shor's algorithm (a dramatic speedup for a problem profoundly relevant to modern electronic commerce) or the broad applicability of Grover's algorithm (a modest yet surprising speedup for the most basic of search problems), recent years have seen a steady stream of new quantum algorithms both for artificial problems that shed light on the power of quantum computation and for problems of genuine practical interest.

In 1996, Grover (1997) gave an algorithm achieving quadratic speedup[3] for the unstructured search problem, the problem of deciding whether a black-box Boolean function has any input that evaluates to 1. Grover's algorithm was subsequently generalized to the framework of amplitude amplification and to counting the number of solutions by Brassard *et al.* (2002). The unstructured search problem is extremely basic, and Grover's algorithm has found application to a wide variety of related problems [see, e.g., Brassard *et al.* (1997); Dürr *et al.* (2004); Ambainis and Špalek (2006)].

The concept of quantum walk, developed by analogy to the classical notion of random walk, has proven to be another broadly useful tool for quantum algorithms. Continuous-time quantum walk was introduced by Farhi and Gutmann (1998) and discrete-time quantum walk was introduced by Watrous (2001b). The continuous-time formulation has been used to demonstrate exponential speedup of quantum over classical computation (Childs *et al.*, 2003, 2007), though it remains to be seen whether these ideas can be applied to a problem of practical interest. However, both continuous- and discrete-time quantum walk have been applied to achieve poly-

[1] In principle, quantum systems evolving according to interactions from a simple initial configuration can be described using fewer parameters, and classical simulations exploiting this idea have been developed [see, e.g., Pérez-García *et al.* (2007)]. But while these ideas are fruitful for simulating some quantum systems, we do not expect them to be efficient for any physically reasonable system—in particular, not for systems capable of performing universal quantum computation. However, we emphasize that there is no unconditional proof that classical simulation of quantum systems requires exponential overhead.

[2] A similar situation occurs for the description of n probabilistic bits by 2^n real-valued probabilities. However, probabilities do not interfere and, contrary to the quantum case, randomized algorithms are not believed to be more powerful than deterministic ones [see, e.g., Impagliazzo and Wigderson (1997)].

[3] Prior to Grover's result it was shown by Bennett *et al.* (1997) that a quadratic speedup for the unstructured search problem is optimal. More generally, for any total Boolean function there can be at most a polynomial separation (in general, at most degree 6) between classical and quantum query complexities (Beals *et al.*, 2001).

nomial speedup for a variety of search problems. Following related work on spatial search (Shenvi *et al.*, 2003; Childs and Goldstone, 2004a, 2004b; Aaronson and Ambainis, 2005; Ambainis *et al.*, 2005), Ambainis (2007) gave an optimal quantum algorithm for the element distinctness problem. This approach was subsequently generalized (Szegedy, 2007; Magniez, Nayak, *et al.*, 2007) and applied to other problems in query complexity, namely, triangle finding (Magniez, Santha, and Szebedy, 2005), checking matrix multiplication (Buhrman and Špalek, 2006), and testing group commutativity (Magniez and Nayak, 2007). Recently, quantum walk has also been applied to give optimal quantum algorithms for evaluating balanced binary game trees (Farhi *et al.*, 2007) and, more generally, Boolean formulas (Ambainis *et al.*, 2007; Reichardt and Špalek, 2008).

A related technique for quantum algorithms is the concept of adiabatic evolution. The quantum adiabatic theorem guarantees that a quantum system in its ground state will remain close to its ground state as the Hamiltonian is changed, provided the change is sufficiently slow, depending on spectral properties of the Hamiltonian [see, for example, Born and Fock (1928); Jansen *et al.* (2007)]. Farhi *et al.* (2000) proposed using adiabatic evolution as an approach to optimization problems. Unfortunately, analyzing this approach is challenging. While it is possible to construct specific cost functions for which specific formulations of adiabatic optimization fail (Fisher, 1992; van Dam *et al.*, 2001; van Dam and Vazirani, 2003; Reichardt, 2004), the performance in general remains poorly understood. Going beyond the setting of optimization problems, adiabatic evolution can simulate a universal quantum computer (Aharonov, van Dam, *et al.*, 2007).

Finally, returning to the original motivation for quantum computation, the vision of Manin and Feynman of quantum computers as quantum simulators has been considerably developed [see, e.g., Lloyd (1996); Wiesner (1996); Zalka (1998); Aspuru-Guzik *et al.* (2005)]. However, it has proven difficult to identify a concrete computational task involving quantum simulation for which the speedup over classical computers can be understood precisely. While it is widely expected that quantum simulation will be one of the major applications of quantum computers, much work remains to be done.

This article is organized as follows. In Sec. II, we give an introduction to the model of quantum computation and the complexity of quantum algorithms. In Sec. III, we introduce the Abelian quantum Fourier transform, and in Sec. IV, we show how this transform can be applied to solve the Abelian hidden subgroup problem, with various applications. In Sec. V, we describe quantum algorithms for problems involving number fields, including the efficient quantum algorithm for solving Pell's equation. In Sec. VI, we introduce the non-Abelian version of the quantum Fourier transform, and in Sec. VII, we discuss the status of the non-Abelian version of the hidden subgroup problem. In Secs. VIII and IX, we describe two approaches to going beyond the hidden subgroup framework, namely, hidden shift problems and hidden nonlinear structure problems, respectively. Finally, in Sec. X, we discuss quantum algorithms for approximating the Jones polynomial and other #P-complete problems.

II. COMPLEXITY OF QUANTUM COMPUTATION

In this section we give an introduction to quantum computers, with particular emphasis on characterizing computational efficiency. For more detailed background, see Preskill (1998a); Nielsen and Chuang (2000); Kitaev *et al.* (2002); Kaye *et al.* (2007).

A. Quantum data

A quantum computer is a device for performing calculations using a quantum mechanical representation of information. Data are stored using quantum bits or *qubits*, the states of which can be represented by ℓ_2-normalized vectors in a complex vector space. For example, we can write the state of n qubits as

$$|\psi\rangle = \sum_{x\in\{0,1\}^n} a_x|x\rangle, \tag{1}$$

where the $a_x \in \mathbb{C}$ satisfy $\Sigma_{x\in\{0,1\}^n}|a_x|^2=1$. We refer to the basis of states $|x\rangle$ as the *computational basis*.

Although we can always suppose that our data are represented using qubits, it is often useful to think of quantum states as storing data more abstractly. For example, given a group G, we write $|g\rangle$ for a computational basis state corresponding to the group element $g\in G$ and

$$|\phi\rangle = \sum_{g\in G} b_g|g\rangle \tag{2}$$

(where $b_g\in\mathbb{C}$ with $\Sigma_{g\in G}|b_g|^2=1$) for an arbitrary superposition over the group. We often implicitly assume that there is some canonical way of concisely representing group elements using bit strings; it is usually unnecessary to make this representation explicit. We use the convention that for any finite set S, the state $|S\rangle$ denotes the normalized uniform superposition of its elements, i.e.,

$$|S\rangle := \frac{1}{\sqrt{|S|}}\sum_{s\in S}|s\rangle. \tag{3}$$

If a quantum computer stores the state $|\psi\rangle$ in one register and the state $|\phi\rangle$ in another, the overall state is given by the tensor product of those two states. This may variously be denoted $|\psi\rangle\otimes|\phi\rangle$, $|\psi\rangle|\phi\rangle$, or $|\psi,\phi\rangle$.

It can be useful to consider statistical mixtures of pure quantum states, represented by *density matrices*. We refer the reader to the references above for further details.

B. Quantum circuits

The allowed operations on pure quantum states are those that map normalized states to normalized states, namely, *unitary operators* U, satisfying $UU^\dagger=U^\dagger U=1$.

When viewed as an $N\times N$ matrix, the rows (and columns) of U form an orthonormal basis of the space $\mathbb{C}^N$.

To have a sensible notion of efficient computation, we require that the unitary operators appearing in a quantum computation are realized by quantum circuits (Deutsch, 1989; Yao, 1993). We are given a set of gates, each of which acts on one or two qubits at a time, meaning that it is a tensor product of a nontrivial one- or two-qubit operator with the identity operator on the remaining qubits. A quantum computation begins in the $|0\cdots0\rangle$ state, applies a sequence of one- and two-qubit gates chosen from the set of allowed gates, and finally reports an outcome obtained by measuring in the computational basis. A circuit is called efficient if it contains a number of gates that is polynomial in the number of qubits the circuit acts on.

In principle, any unitary operator on n qubits can be implemented using only one- and two-qubit gates (DiVincenzo, 1995). Thus we say that the set of all one- and two-qubit gates is (exactly) universal. Of course, some unitary operators take many more one- and two-qubit gates to realize than others, and, indeed, a simple counting argument shows that most unitary operators on n qubits can only be realized using an exponentially large circuit (Knill, 1995).

In general, we are content with circuits that give good approximations of our desired unitary transformations. We say that a circuit with gates $U_1,U_2,\ldots,U_t$ approximates U with precision ε if $\|U-U_t\cdots U_2U_1\|\leqslant\varepsilon$, where $\|\cdot\|$ denotes the operator norm, i.e., the largest singular value. We call a set of elementary gates universal if any unitary operator on a fixed number of qubits can be approximated to precision ε using poly$(\log\frac{1}{\varepsilon})$ elementary gates. It turns out that there are finite sets of gates that are universal (Boykin *et al.*, 2000): for example, the set $\{H,T,\Lambda(X)\}$ with

$$H \coloneqq \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}, \quad T \coloneqq \begin{pmatrix}e^{i\pi/8} & 0\\ 0 & e^{-i\pi/8}\end{pmatrix}, \tag{4}$$

$$\Lambda(X) \coloneqq \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{pmatrix}. \tag{5}$$

There are situations in which a set of gates is effectively universal even though it cannot actually approximate any unitary operator on n qubits. For example, the gate set $\{H,T^2,\Lambda(X),\Lambda^2(X)\}$, where $\Lambda^2(X)$ denotes the Toffoli gate $[\Lambda^2(X)|xyz\rangle=|xyz\rangle$ for $xy\in\{00,01,10\}$ and $\Lambda^2(X)|11z\rangle=|11\bar{z}\rangle]$ is universal (Kitaev, 1997) but only if we allow the use of ancilla qubits (qubits that start and end in the $|0\rangle$ state). Similarly, the gate set $\{H,\Lambda^2(X)\}$ is universal in the sense that, with ancillas, it can approximate any *orthogonal* transformation (Aharonov, 2003; Shi, 2003). It clearly cannot approximate complex unitary matrices since the entries of H and $\Lambda^2(X)$ are real, but the effect of arbitrary unitary transformations can be simulated using orthogonal ones by simulating the real and imaginary parts separately (Bernstein and Vazirani, 1993; Rudolph and Grover, 2002).

One might wonder whether some universal gate sets are better than others. It turns out that the answer is essentially no: a unitary operator that can be realized efficiently with one set of one- and two-qubit gates can also be realized efficiently with another such set. This is a consequence of the Solovay-Kitaev theorem (Kitaev, 1997; Solovay, 2000; Harrow *et al.*, 2002).

Theorem 1. Fix two gate sets that allow universal quantum computation and that are closed under taking inverses. Then any t-gate circuit using the first gate set can be implemented with error at most ε using a circuit of $t\cdot$poly$[\log(t/\varepsilon)]$ gates from the second gate set. Furthermore, there is an efficient classical algorithm for finding this circuit.

This means we can view a simple finite gate set, such as $\{H,T,\Lambda(X)\}$, as equivalent to an infinite gate set, such as the set of all two-qubit gates. A finite gate set is needed both for fault tolerance (Sec. II.E) and for the concept of uniformly generated circuits (see footnote 4).

Note that to implement unitary operators exactly, the notion of efficiency might depend on the allowed gates [see, e.g., Mosca and Zalka (2004)] so we usually restrict our attention to quantum computation with bounded error.

In principle, one can construct quantum circuits adaptively, basing the choices of gates on the outcomes of intermediate measurements. We may also discard quantum data in the course of a circuit. In general, the possible operations on mixed quantum states correspond to completely positive, trace preserving maps on density matrices. Again, we refer the reader to the aforementioned references for more details.

C. Reversible computation

Unitary matrices are invertible: in particular, $U^{-1}=U^\dagger$. Thus any unitary transformation is a reversible operation. This may seem at odds with how we often define classical circuits, using irreversible gates such as AND and OR. But any classical computation can be made reversible by replacing each irreversible gate $x\mapsto g(x)$ by the reversible gate $(x,y)\mapsto(x,y\oplus g(x))$, where $\oplus$ denotes bitwise addition mod 2. Applying this gate to the input $(x,0)$ produces $(x,g(x))$. By storing all intermediate steps of the computation, we make it reversible (Bennett, 1973).

On a quantum computer, storing all intermediate computational steps could present a problem since two identical results obtained via distinct computational histories would not be able to interfere. However, there is an easy way to remove the accumulated information. After performing the classical computation with reversible gates, we simply copy the answer into an ancilla register and then perform the computation in reverse. Thus we can implement the map $(x,y)\mapsto(x,y\oplus f(x))$ even when f is a complicated circuit consisting of many gates.

Using this trick, any computation that can be performed efficiently on a classical computer can be performed efficiently on a quantum computer even on a superposition of computational basis states. In other words, if we can efficiently implement the map $x \mapsto f(x)$ on a classical computer, we can efficiently perform the transformation

$$\sum_x a_x|x,y\rangle \mapsto \sum_x a_x|x,y \oplus f(x)\rangle \tag{6}$$

on a quantum computer. Note that this does not necessarily mean we can efficiently perform the transformation

$$\sum_x a_x|x\rangle \mapsto \sum_x a_x|f(x)\rangle \tag{7}$$

even if the function f is bijective.

D. Quantum complexity theory

We say that an algorithm for a problem is efficient if the circuit describing it uses a number of gates that is polynomial in the input size, the number of bits needed to write down the input.[4] For example, if the input is an integer mod N, the input size is $\lceil \log_2 N \rceil$.

With a quantum computer, as with a randomized (or noisy) classical computer, the final result of a computation may not be correct with certainty. Instead, we are typically content with an algorithm that can produce the correct answer with high enough probability. To solve a decision problem, it suffices to give an algorithm with success probability bounded above $1/2$ (say, at least $2/3$) since we can repeat the computation many times and take a majority vote to make the probability of outputting an incorrect answer arbitrarily small. Similarly, if we can check whether a given solution is correct, it suffices to output the correct answer with probability $\Omega(1)$.[5]

It is common practice to characterize the difficulty of computational problems using *complexity classes* [see, e.g., Papadimitriou (1994)]. Typically, these classes contain decision problems, problems with a "yes" or "no" answer. (Such a problem is conventionally formulated as deciding whether a string over some finite alphabet is in a given *language*; formally, a complexity class is a set of languages.) For example, the problems that can be decided in polynomial time on a deterministic classical computer belong to the class P, on a probabilistic classical computer with error at most $1/3$ to the class BPP, and on a quantum computer with error at most $1/3$ to the class BQP. Clearly, P⊆BPP⊆BQP. The central problem of quantum algorithms can be viewed as trying to understand what problems are in BQP but not in P (or BPP).

Whereas the classes P, BPP, and BQP all attempt to characterize modes of computation that could be carried out in practice, *computational complexity theory* is also concerned with more abstract classes that characterize other aspects of computation. For example, the class NP corresponds to those decision problems for which a "yes" answer can be verified in polynomial time on a classical computer, given a succinct proof. It is widely believed that P≠NP and, indeed, that NP⊈BQP (though it is also plausible that BQP⊈NP), but proving this appears to be a challenging problem [see, e.g., the survey of quantum complexity by Watrous (2009)]. Indeed, it seems almost as difficult just to prove P≠PSPACE, where PSPACE denotes the class of problems that can be decided by a deterministic classical computer running in polynomial space. Since BQP⊆PSPACE (Bernstein and Vazirani, 1997) [i.e., any computation that can be performed on a quantum computer in polynomial time can be performed on a classical computer with polynomial memory—indeed, even stronger such results are known (Adleman *et al.*, 1997; Bernstein and Vazirani, 1997; Fortnow and Rogers, 1998)], we expect it will be hard to prove P≠BQP. Instead, we try to find efficient quantum algorithms for problems that at least appear to be hard for classical computers.

While most complexity classes contain decision problems, some classes describe the complexity of computing non-Boolean functions. For example, the class #P characterizes the complexity of counting the number of "yes" solutions to a problem in NP.

Alternatively, instead of considering natural computational problems (in which the input is a string), we sometimes work in the setting of *query complexity*. Here the input is a black-box transformation (or *oracle*)—which in the quantum setting is given as a unitary transformation as in Eq. (6)—and our goal is to discover some property of the transformation by querying it as few times as possible. For example, in Simon's problem (Simon, 1997), we are given a black box for a transformation $f:\{0,1\}^n \to S$ satisfying $f(x)=f(y)$ if and only if $y \in \{x, x \oplus t\}$ for some unknown $t \in \{0,1\}^n$ and the goal is to learn t.

The query model facilitates proving lower bounds: it is often tractable to establish that many queries must be used to solve a given black-box problem, whereas it is generally hard to show that many gates are required to

[4]Strictly speaking, we want the circuits for solving instances of a problem of different sizes to be simply related to one another. Given the ability to choose an arbitrary circuit for each input size, we could have circuits computing uncomputable functions (i.e., functions that a Turing machine could not compute). Thus we require our circuits to be uniformly generated: say, that there exists a fixed (classical) Turing machine that, given a tape containing the symbol "1" n times, outputs a description of the nth circuit in time poly(n).

[5]We use standard big-O notation here, where $f=O(g)$ if there exist positive constants c,y such that $|f(x)| \leqslant c|g(x)|$ for all $x \geqslant y$, $f=\Omega(g)$ if $g=O(f)$, and $f=\Theta(g)$ if both $f=O(g)$ and $f=\Omega(g)$. The expression $\Omega(1)$ thus represents a function asymptotically lower bounded by an unspecified positive constant. We write $f=o(g)$ to denote that $\lim_{x\to\infty} f(x)/g(x)=0$. To convey that a function f is bounded from above by a polynomial in the function g, we write $f=\mathrm{poly}(g)$, which could also be written as $f=g^{O(1)}$.

compute some explicit function. Indeed, this article describes numerous examples of black-box problems that can be solved in polynomial time on a quantum computer, but that provably require exponentially many queries on a randomized classical computer. Of course, if we find an efficient algorithm for a problem in query complexity, and if we are provided with an explicit efficient circuit realizing the black-box transformation, then we have an efficient algorithm for a natural computational problem. We emphasize, however, that lower bounds in the query model no longer apply when the black box is thus replaced by a transparent one. For example, Shor's factoring algorithm (Sec. IV.E) proceeds by solving a problem in query complexity which is provably hard for classical computers. Nevertheless, it is an open question whether factoring is classically hard since there might be a fast classical algorithm that does not work by solving the query problem.

E. Fault tolerance

With any real computer, operations cannot be done perfectly. Quantum gates and measurements may be performed imprecisely, and errors may happen even to stored data that are not being manipulated. Fortunately, there are protocols for dealing with faults that occur during the execution of a quantum computation. Specifically, the fault-tolerant threshold theorem states that as long as the noise level is below some threshold (depending on the noise model and the architecture of the quantum computer but typically in the range of 10^{-2}–10^{-4}), an arbitrarily long computation can be performed with arbitrarily small error (Knill *et al.*, 1996, 1997; Shor, 1996; Kitaev, 1997; Preskill, 1998b; Aharonov and Ben-Or, 2008). Throughout we implicitly assume that fault-tolerant protocols have been applied, so that we effectively have a perfectly functioning quantum computer.

III. ABELIAN QUANTUM FOURIER TRANSFORM

A. Fourier transforms over finite Abelian groups

For the group $\mathbb{Z}/N\mathbb{Z}$, the group of integers mod N under addition (see Appendix A), the quantum Fourier transform (QFT) is a unitary operation $F_{\mathbb{Z}/N\mathbb{Z}}$. Its effect on a basis state $|x\rangle$ for any $x \in \mathbb{Z}/N\mathbb{Z}$ is

$$|x\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{y \in \mathbb{Z}/N\mathbb{Z}} \omega_N^{xy} |y\rangle, \tag{8}$$

where $\omega_N := e^{2\pi i/N}$ denotes a primitive Nth root of unity.

More generally, a finite Abelian group G has $|G|$ distinct one-dimensional irreducible representations (or irreducible characters), $\psi \in \hat{G}$. These are functions $\psi: G \to \mathbb{C}$ with $\psi(a+b) = \psi(a)\psi(b)$ for all $a, b \in G$, using additive notation for the group operation of G (see Appendix B for further details). The quantum Fourier transform F_G over G acts as

$$|x\rangle \mapsto \frac{1}{\sqrt{|G|}} \sum_{\psi \in \hat{G}} \psi(x) |\psi\rangle \tag{9}$$

for each $x \in G$.

For example, the group $(\mathbb{Z}/N\mathbb{Z}) \times (\mathbb{Z}/N\mathbb{Z})$ has N^2 irreducible representations defined by $\psi_{y_1,y_2}: (x_1, x_2) \mapsto \omega_N^{x_1 y_1 + x_2 y_2}$ for all $y_1, y_2 \in \mathbb{Z}/N\mathbb{Z}$; hence its quantum Fourier transform $F_{(\mathbb{Z}/N\mathbb{Z}) \times (\mathbb{Z}/N\mathbb{Z})}$ acts as

$$|x_1, x_2\rangle \mapsto \frac{1}{N} \sum_{y_1, y_2 \in \mathbb{Z}/N\mathbb{Z}} \omega_N^{x_1 y_1 + x_2 y_2} |y_1, y_2\rangle \tag{10}$$

for all $x_1, x_2 \in \mathbb{Z}/N\mathbb{Z}$. In this example, $F_{(\mathbb{Z}/N\mathbb{Z}) \times (\mathbb{Z}/N\mathbb{Z})}$ can be written as the tensor product $F_{\mathbb{Z}/N\mathbb{Z}} \otimes F_{\mathbb{Z}/N\mathbb{Z}}$. In general, according to the fundamental theorem of finite Abelian groups, any finite Abelian group G can be expressed as a direct product of cyclic subgroups of prime power order $G \cong (\mathbb{Z}/p_1^{r_1}\mathbb{Z}) \times \cdots \times (\mathbb{Z}/p_k^{r_k}\mathbb{Z})$, and the QFT over G can be written as the tensor product of QFTs $F_{\mathbb{Z}/p_1^{r_1}\mathbb{Z}} \otimes \cdots \otimes F_{\mathbb{Z}/p_k^{r_k}\mathbb{Z}}$.

The Fourier transform F_G is useful for exploiting symmetry with respect to G. Consider the operator P_s that adds $s \in G$, defined by $P_s|x\rangle = |x+s\rangle$ for any $x \in G$. This operator is diagonal in the Fourier basis: we have

$$F_G P_s F_G^\dagger = \sum_{\psi \in \hat{G}} \psi(s) |\psi\rangle\langle\psi|. \tag{11}$$

Thus, measurements in the Fourier basis produce the same statistics for a pure state $|\phi\rangle$ and its shift $P_s|\phi\rangle$. Equivalently, a G-invariant mixed state is diagonalized by F_G.

B. Efficient quantum circuit for the QFT over $\mathbb{Z}/2^n\mathbb{Z}$

To use the Fourier transform over G as part of an efficient quantum computation, we must implement it by a quantum circuit of size poly$(\log|G|)$. This can be done for any finite Abelian group (Cleve, 1994; Coppersmith, 1994; Kitaev, 1995; Barenco *et al.*, 1996; Shor, 1997; Hales and Hallgren, 2000). In this section we explain a construction for the case of the group $\mathbb{Z}/2^n\mathbb{Z}$, following the presentation of Cleve *et al.* (1998).

Transforming from the basis of states $\{|x\rangle : x \in G\}$ to the basis $\{|\psi\rangle : \psi \in \hat{G}\}$, the matrix representation of the Fourier transformation over $\mathbb{Z}/N\mathbb{Z}$ is

$$F_{\mathbb{Z}/N\mathbb{Z}} = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_N & \omega_N^2 & \cdots & \omega_N^{N-1} \\ 1 & \omega_N^2 & \omega_N^4 & \cdots & \omega_N^{2N-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_N^{N-1} & \omega_N^{2N-2} & \cdots & \omega_N^{(N-1)(N-1)} \end{pmatrix}. \tag{12}$$

More succinctly,

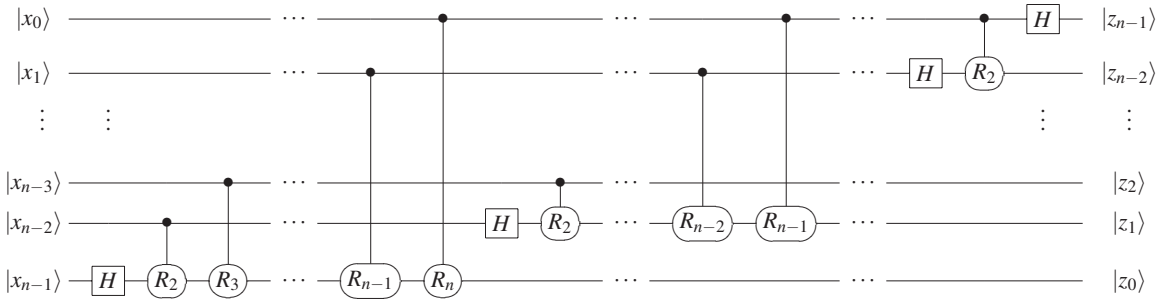


FIG. 1. An efficient [size $O(n^2)$] quantum circuit for the quantum Fourier transform over $\mathbb{Z}/2^n\mathbb{Z}$. Note that the order of the n output bits $z_0,\ldots,z_{n-1}$ is reversed, as compared with the order of the n input bits $x_0,\ldots,x_{n-1}$.

$$F_{\mathbb{Z}/N\mathbb{Z}} = \frac{1}{\sqrt{N}} \sum_{x,y\in\mathbb{Z}/N\mathbb{Z}} \omega_N^{xy} |y\rangle\langle x|, \tag{13}$$

where $|y\rangle$ represents the basis state corresponding to the character ψ_y with $\psi_y(x)=\omega_N^{xy}$. It is straightforward to verify that $F_{\mathbb{Z}/N\mathbb{Z}}$ is indeed a unitary transformation, i.e., that $F_{\mathbb{Z}/N\mathbb{Z}}F_{\mathbb{Z}/N\mathbb{Z}}^\dagger = F_{\mathbb{Z}/N\mathbb{Z}}^\dagger F_{\mathbb{Z}/N\mathbb{Z}} = 1$.

We assume now that $N=2^n$ and represent the integer $x\in\mathbb{Z}/N\mathbb{Z}$ by n bits $x_0,x_1,\ldots,x_{n-1}$, where $x=\Sigma_{j=0}^{n-1}2^j x_j$. The Fourier transform of $|x\rangle$ can then be written as the tensor product of n qubits since

$$\begin{aligned} F_{\mathbb{Z}/2^n\mathbb{Z}}|x\rangle &= \frac{1}{\sqrt{2^n}} \sum_{y\in\{0,1\}^n} \omega_{2^n}^{x(\Sigma_{j=0}^{n-1}2^j y_j)} |y_0,\ldots,y_{n-1}\rangle \\ &= \frac{1}{\sqrt{2^n}} \bigotimes_{j=0}^{n-1} \sum_{y_j\in\{0,1\}} e^{2\pi i x y_j/2^{n-j}} |y_j\rangle \\ &= \bigotimes_{j=0}^{n-1} \frac{|0\rangle + \exp\left(2\pi i \sum_{k=0}^{n-1} 2^{j+k-n} x_k\right)|1\rangle}{\sqrt{2}} \\ &=: \bigotimes_{j=0}^{n-1} |z_j\rangle. \end{aligned} \tag{14}$$

Now, because $\exp(2\pi i 2^s x_k)=1$ for all integers $s\geq 0$, we see that the jth output qubit is

$$|z_j\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i(2^{j-n}x_0 + 2^{j+1-n}x_1 + \cdots + 2^{-1}x_{n-1-j})}|1\rangle) \tag{15}$$

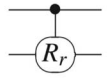
and hence only depends on the $n-j$ input bits $x_0,\ldots,x_{n-1-j}$.

To describe a quantum circuit that implements the Fourier transform, we define the single-qubit phase rotation

$$R_r := \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^r} \end{pmatrix} \simeq \text{—}(R_r)\text{—} \tag{16}$$

and the two-qubit controlled rotation

$$\Lambda(R_r) := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{2\pi i/2^r} \end{pmatrix} \simeq \tag{17}$$



acting symmetrically on a and $b\in\{0,1\}$ as $\Lambda(R_r)|a,b\rangle = e^{2\pi i ab/2^r}|a,b\rangle$. The circuit shown in Fig. 1 uses $\binom{n}{2}$ of these gates together with n Hadamard gates to exactly implement the quantum Fourier transform over $\mathbb{Z}/2^n\mathbb{Z}$.

In this circuit, there are many rotations by small angles that do not significantly affect the final result. By simply omitting the gates $\Lambda(R_r)$ with $r=\Omega(\log n)$, we obtain a circuit of size $O(n\log n)$ [instead of $O(n^2)$ for the original circuit] that implements the QFT with precision $1/\mathrm{poly}(n)$ (Coppersmith, 1994).

C. Phase estimation and the QFT over any finite Abelian group

Aside from being directly applicable to quantum algorithms, such as Shor's algorithm, the QFT over $\mathbb{Z}/2^n\mathbb{Z}$ provides a useful quantum computing primitive called *phase estimation* (Kitaev, 1995; Cleve *et al.* 1998). In the phase estimation problem, we are given a unitary operator U (either as an explicit circuit or as a black box that applies a controlled-U^x operation for integer values of x). We are also given a state $|\phi\rangle$ that is an eigenvector of U, namely, $U|\phi\rangle = e^{i\phi}|\phi\rangle$ for some $\phi\in\mathbb{R}$. The goal is to output an estimate of ϕ to some desired precision. (Of course, we can also apply the procedure to a general state $|\psi\rangle$; by linearity, we obtain each value ϕ with probability $|\langle\phi|\psi\rangle|^2$.)

The procedure for phase estimation is straightforward:

Algorithm 1 (Phase estimation).

Input: Eigenstate $|\phi\rangle$ (with eigenvalue $e^{i\phi}$) of a given unitary operator U.

Problem: Produce an n-bit estimate of ϕ.

(1) Prepare the quantum computer in the state

$$\frac{1}{\sqrt{2^n}} \sum_{x\in\mathbb{Z}/2^n\mathbb{Z}} |x\rangle\otimes|\phi\rangle. \tag{18}$$

(2) Apply the unitary operator

$$\sum_{x\in\mathbb{Z}/2^n\mathbb{Z}} |x\rangle\langle x| \otimes U^x, \tag{19}$$

giving the state

$$\frac{1}{\sqrt{2^n}} \sum_{x\in\mathbb{Z}/2^n\mathbb{Z}} e^{i\phi x}|x\rangle \otimes |\phi\rangle. \tag{20}$$

(3) Apply an inverse Fourier transform on the first register, giving

$$\frac{1}{2^n} \sum_{x,y\in\mathbb{Z}/2^n\mathbb{Z}} \omega_{2^n}^{x[(2^n/2\pi)\phi-y]}|y\rangle \otimes |\phi\rangle. \tag{21}$$

(4) Measure the first register of the resulting state in the computational basis.

If the binary expansion of $\phi/2\pi$ terminates after at most n bits, then the result is guaranteed to be the binary expansion of $\phi/2\pi$. In general, we obtain a good approximation with high probability (Cleve *et al.*, 1998). (The relevant calculation appears in Sec. IV.D for the case where $\phi\in\mathbb{Q}$; that same calculation works for any $\phi\in\mathbb{R}$.) The optimal way of estimating the unknown phase has been analyzed by van Dam *et al.* (2007), but the above method is sufficient for our purposes.

The complexity of Algorithm 1 can depend on the form of the unitary operator U. If we are only given a black box for the controlled-U gate, then there may be no better way to implement the controlled-U^x operation than by performing a controlled-U gate x times, so that the running time is $\Theta(2^n)$ (i.e., approximately the inverse of the desired precision). On the other hand, if it is possible to implement Eq. (19) in poly(n) time (say, using repeated squaring) then phase estimation can be performed in poly(n) time.

One useful application of phase estimation is to implement the QFT [Eq. (13)] over an arbitrary cyclic group $\mathbb{Z}/N\mathbb{Z}$ (Kitaev, 1995). The circuit presented in the previous section only works when N is a power of 2 (or, with a slight generalization, a power of some other fixed integer). But the following simple technique can be used to realize $F_{\mathbb{Z}/N\mathbb{Z}}$ (approximately) using phase estimation. [While this approach is conceptually simple, it is possible to implement the QFT over a cyclic group more efficiently; see Hales and Hallgren (2000).]

Our goal is to perform the transformation that maps $|x\rangle\mapsto|\hat{x}\rangle$, where $|\hat{x}\rangle := F_{\mathbb{Z}/N\mathbb{Z}}|x\rangle$ denotes a Fourier basis state. By linearity, if the transformation acts correctly on a basis, it acts correctly on all states. It is straightforward to perform the transformation $|x,0\rangle\mapsto|x,\hat{x}\rangle$ (create a uniform superposition $\Sigma_{y\in\mathbb{Z}/N\mathbb{Z}}|y\rangle/\sqrt{N}$ in the second register and apply the controlled phase shift $|x,y\rangle\mapsto\omega_N^{xy}|x,y\rangle$), but it remains to erase the first register.

Consider the unitary operator P_1 that adds 1 mod N, i.e., $P_1|x\rangle=|x+1\rangle$ for any $x\in\mathbb{Z}/N\mathbb{Z}$. According to Eq. (11), the eigenstates of this operator are precisely the Fourier basis states $|\hat{x}\rangle$, with eigenvalues ω_N^x. Thus, using phase estimation on P_1 [with $n=O(\log N)$ bits of precision], we can approximate the transformation $|\hat{x},0\rangle\mapsto|\hat{x},x\rangle$. Reversing this operation, we can erase $|x\rangle$, giving the desired QFT. Note that we can perform P_1^x in poly(log N) steps even when x is exponentially large in log N, so the resulting procedure is indeed efficient.

Given the Fourier transform over $\mathbb{Z}/N\mathbb{Z}$, it is straightforward to implement the QFT over an arbitrary finite Abelian group using the decomposition of the group into cyclic factors, as discussed in Sec. III.A.

If gates can be performed in parallel, it is possible to perform the QFT much more quickly, using only $O(\log\log N)$ time steps (Cleve and Watrous, 2000; Hales, 2002).

D. The QFT over a finite field

The elements of the finite field $\mathbb{F}_q$, where $q=p^m$ is a power of a prime number p, form an Abelian group under addition (see Appendix A), and the QFT over this group has many applications. If q is prime, then $\mathbb{F}_q=\mathbb{Z}/q\mathbb{Z}$, so the QFT over $\mathbb{F}_q$ is straightforward. More generally, as an additive group $\mathbb{F}_q\cong(\mathbb{Z}/p\mathbb{Z})^m$, so in principle the QFT over $\mathbb{F}_q$ could be defined using an explicit isomorphism to $(\mathbb{Z}/p\mathbb{Z})^m$. However, it is often more convenient to define $F_{\mathbb{F}_q}$ in terms of the (absolute) trace, the linear function $\mathrm{Tr}:\mathbb{F}_q\to\mathbb{F}_p$ defined by

$$\mathrm{Tr}(x) := x + x^p + x^{p^2} + \cdots + x^{p^{m-1}}. \tag{22}$$

One can show that the functions $\psi_y:\mathbb{F}_q\to\mathbb{C}$ defined by

$$\psi_y(x) = \omega_p^{\mathrm{Tr}(xy)} \tag{23}$$

for each $y\in\mathbb{F}_q$ form a complete set of additive characters of $\mathbb{F}_q$. Thus, the QFT over $\mathbb{F}_q$ can be written

$$F_{\mathbb{F}_q} = \frac{1}{\sqrt{q}} \sum_{x,y\in\mathbb{F}_q} \omega_p^{\mathrm{Tr}(xy)}|y\rangle\langle x|. \tag{24}$$

This definition is preferred over other possible choices because it commutes with the permutation $|z\rangle\mapsto|z^p\rangle$ implementing the Frobenius automorphism and hence respects the multiplicative structure of $\mathbb{F}_q$.

IV. ABELIAN HIDDEN SUBGROUP PROBLEM

A. Period finding over $\mathbb{Z}/N\mathbb{Z}$

Suppose we are given a function over the integers $0,1,\ldots,N-1$ that is periodic with period r. Further, suppose that this function never takes the same value twice within the fundamental period (i.e., it is *injective* within each period). In other words, the function $f:\mathbb{Z}/N\mathbb{Z}\to S$ satisfies

$$f(x)=f(y) \text{ if and only if } (x-y)/r\in\mathbb{Z} \tag{25}$$

for all $x,y\in\mathbb{Z}/N\mathbb{Z}$. Notice that this can only be the case if r divides N, so that f can have exactly N/r periods.

If we know N, then we can find the period r efficiently using the quantum Fourier transform over the additive group $\mathbb{Z}/N\mathbb{Z}$. We represent each element $x\in\mathbb{Z}/N\mathbb{Z}$ uniquely as an integer $x\in\{0,\ldots,N-1\}$. Similarly, the ir-

reducible representations $\psi:\mathbb{Z}/N\mathbb{Z}\to\mathbb{C}$ can be labeled by integers $y\in\{0,\ldots,N-1\}$, namely, with $\psi_y(x)=e^{2\pi ixy/N}$. The following algorithm solves the period-finding problem.

Algorithm 2 (*Period finding over* $\mathbb{Z}/N\mathbb{Z}$).

Input: A black box $f:\mathbb{Z}/N\mathbb{Z}\to S$ satisfying Eq. (28) for some unknown $r\in\mathbb{Z}/N\mathbb{Z}$, where r divides N.

Problem: Determine r.

(1) Create the uniform superposition

$$|\mathbb{Z}/N\mathbb{Z}\rangle=\frac{1}{\sqrt{N}}\sum_{x\in\mathbb{Z}/N\mathbb{Z}}|x\rangle \tag{26}$$

of all elements of $\mathbb{Z}/N\mathbb{Z}$ [recall the notation Eq. (3)]. For example, this can be done by applying the Fourier transform over $\mathbb{Z}/N\mathbb{Z}$ to the state $|0\rangle$.

(2) Query the function f in an ancilla register, giving

$$\frac{1}{\sqrt{N}}\sum_{x\in\mathbb{Z}/N\mathbb{Z}}|x,f(x)\rangle. \tag{27}$$

(3) At this point, if we were to measure the ancilla register, the first register would be left in a superposition of those $x\in\mathbb{Z}/N\mathbb{Z}$ consistent with the observed function value. By the periodicity of f, this state would be of the form

$$\sqrt{\frac{r}{N}}\sum_{j=0}^{N/r-1}|s+jr\rangle \tag{28}$$

for some unknown offset $s\in\{0,\ldots,r-1\}$ occurring uniformly at random, corresponding to the uniformly random observed function value $f(s)$. Since we will not use this function value, there is no need to explicitly measure the ancilla; ignoring the second register results in the same statistical description. Thus, we may simply discard the ancilla, giving a mixed quantum state or, equivalently, a random pure state.

(4) Apply the Fourier transform over $\mathbb{Z}/N\mathbb{Z}$, giving

$$\sqrt{\frac{r}{N}}\sum_{y\in\mathbb{Z}/N\mathbb{Z}}\sum_{j=0}^{N/r-1}\omega_N^{(s+jr)y}|y\rangle. \tag{29}$$

By the identity

$$\sum_{j=0}^{M-1}\omega_M^{jy}=M\delta_{j,y \bmod M} \tag{30}$$

(applied with $M=N/r$, so $\omega_N^{jry}=\omega_M^{jy}$), only the values $y\in\{0,N/r,2N/r,\ldots,(r-1)N/r\}$ experience constructive interference, and Eq. (29) equals

$$\frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}\omega_r^{sk}|kN/r\rangle. \tag{31}$$

(5) Measure this state in the computational basis, giving some integer multiple kN/r of N/r. Dividing this integer by N gives the fraction k/r, which, when reduced to lowest terms, has $r/\gcd(r,k)$ as its denominator.

(6) Repeating the above gives a second denominator $r/\gcd(r,k')$. If k and k' are relatively prime, the least common multiple of $r/\gcd(r,k)$ and $r/\gcd(r,k')$ is r. The probability of this happening is at least $\Pi_{p\text{ prime}}(1-1/p^2)=6/\pi^2\approx 0.61$, so the algorithm succeeds with constant probability.

B. Computing discrete logarithms

Let $C=\langle g\rangle$ be a cyclic group generated by an element g, with the group operation written multiplicatively. Given an element $x\in C$, the discrete logarithm of x in C with respect to g, denoted $\log_g x$, is the smallest nonnegative integer ℓ such that $g^\ell=x$. The discrete logarithm problem is the problem of calculating $\log_g x$ given g and x. (Notice that for *additive* groups such as $G=\mathbb{Z}/p\mathbb{Z}$, the discrete logarithm represents division: $\log_g x=x/g \bmod p$.)

1. Discrete logarithms and cryptography

Classically, the discrete logarithm seems like a good candidate for a one-way function. We can efficiently compute g^ℓ, even if ℓ is exponentially large (in $\log|C|$), by repeated squaring. But given x, it is not immediately clear how to compute $\log_g x$ without checking exponentially many possibilities.

The apparent hardness of the discrete logarithm problem is the basis of the Diffie-Hellman key exchange protocol (Diffie and Hellman, 1976), the earliest published public-key cryptographic protocol. The goal of key exchange is for two distant parties, Alice and Bob, to agree on a secret key using only an insecure public channel. The Diffie-Hellman protocol works as follows:

(1) Alice and Bob publicly agree on a large prime p and an integer g of high order. For simplicity, suppose they choose a g for which $\langle g\rangle=(\mathbb{Z}/p\mathbb{Z})^\times$ (i.e., a primitive root mod p). (In general, finding such a g might be hard, but it can be done efficiently given certain restrictions on p.)

(2a) Alice chooses some $a\in\mathbb{Z}/(p-1)\mathbb{Z}$ uniformly at random. She computes $A:=g^a \bmod p$ and sends the result to Bob (keeping a secret).

(2b) Bob chooses some $b\in\mathbb{Z}/(p-1)\mathbb{Z}$ uniformly at random. He computes $B:=g^b \bmod p$ and sends the result to Alice (keeping b secret).

(3a) Alice computes $K:=B^a=g^{ab} \bmod p$.

(3b) Bob computes $K=A^b=g^{ab} \bmod p$.

At the end of the protocol, Alice and Bob share a key K, and an eavesdropper Eve has only seen p, g, A, and B.

The security of the Diffie-Hellman protocol relies on the assumption that discrete logarithms are hard to compute. Clearly, if Eve can compute discrete logarithms, she can recover a and b and hence the key. But it is widely believed that the discrete logarithm problem is difficult for classical computers. The best known algorithms for general groups, such as Pollard's rho algorithm and the baby-step giant-step algorithm, run in time $O(\sqrt{|C|})$. For particular groups, it may be possible to do better: for example, over $(\mathbb{Z}/p\mathbb{Z})^\times$ with p prime,

the number field sieve is conjectured to compute discrete logarithms in time $2^{O((\log p)^{1/3}(\log\log p)^{2/3})}$ (Gordon, 1993) [whereas the best known, rigorously analyzed, algorithms run in time $2^{O(\sqrt{\log p \log\log p})}$ (Pomerance, 1987)]; but this is still superpolynomial in $\log p$. It is suspected that breaking the Diffie-Hellman protocol is essentially as hard as computing the discrete logarithm.[6]

This protocol by itself only provides a means of exchanging a secret key, not of sending private messages. However, Alice and Bob can subsequently use their shared key in a symmetric encryption protocol to communicate securely. The ideas behind the Diffie-Hellman protocol can also be used to directly create public-key cryptosystems (similar in spirit to the widely used RSA cryptosystem), such as the ElGamal protocol; see, e.g., Menezes *et al.* (1996), Buchmann (2004).

2. Shor's algorithm for computing discrete logarithms

Although the problem appears to be difficult for classical computers, quantum computers can calculate discrete logarithms efficiently. Recall that we are given some element x of a cyclic group $C=\langle g\rangle$ and we want to calculate $\log_g x$, the smallest non-negative integer ℓ such that $g^\ell = x$.

For simplicity, assume that the order of the group, $N:=|C|$, is known. For example, if $C=(\mathbb{Z}/p\mathbb{Z})^\times$, then we know $N=p-1$. If we do not know N, we can determine it efficiently using Shor's algorithm for period finding over $\mathbb{Z}$, discussed in Sec. IV.D. We also assume that $x\neq g$ (i.e., $\log_g x\neq 1$) since it is easy to check this.

Shor's (1997) algorithm for computing discrete logarithms works as follows.

Algorithm 3 (*Discrete logarithm*).

Input: A cyclic group $C=\langle g\rangle$ and an element $x\in C$.

Problem: Calculate $\log_g x$.

(1) If necessary, using the period-finding algorithm of Sec. IV.D, determine the order $N=|C|$.

(2) Create the uniform superposition

$$|\mathbb{Z}/N\mathbb{Z}\times\mathbb{Z}/N\mathbb{Z}\rangle = \frac{1}{N}\sum_{\alpha,\beta\in\mathbb{Z}/N\mathbb{Z}}|\alpha,\beta\rangle \tag{32}$$

over all elements of the additive Abelian group $\mathbb{Z}/N\mathbb{Z}\times\mathbb{Z}/N\mathbb{Z}$.

(3) Define a function $f:\mathbb{Z}/N\mathbb{Z}\times\mathbb{Z}/N\mathbb{Z}\to C$ as follows:

$$f(\alpha,\beta) = x^\alpha g^\beta. \tag{33}$$

Compute this function in an ancilla register, giving

$$\frac{1}{N}\sum_{\alpha,\beta\in\mathbb{Z}/N\mathbb{Z}}|\alpha,\beta,f(\alpha,\beta)\rangle. \tag{34}$$

(4) Discard the ancilla register.[7] Since $f(\alpha,\beta)=g^{\alpha\log_g x+\beta}$, where f is constant on the lines

$$L_\gamma := \{(\alpha,\beta)\in(\mathbb{Z}/N\mathbb{Z})^2 : \alpha\log_g x+\beta=\gamma\}, \tag{35}$$

so the remaining state is a uniform superposition over group elements consistent with a uniformly random unknown $\gamma\in\mathbb{Z}/N\mathbb{Z}$, namely,

$$|L_\gamma\rangle = \frac{1}{\sqrt{N}}\sum_{\alpha\in\mathbb{Z}/N\mathbb{Z}}|\alpha,\gamma-\alpha\log_g x\rangle. \tag{36}$$

(5) Now we can exploit the symmetry of the quantum state by performing a QFT over $\mathbb{Z}/N\mathbb{Z}\times\mathbb{Z}/N\mathbb{Z}$, giving

$$\begin{aligned}&\frac{1}{N^{3/2}}\sum_{\alpha,\mu,\nu\in\mathbb{Z}/N\mathbb{Z}}\omega_N^{\mu\alpha+\nu(\gamma-\alpha\log_g x)}|\mu,\nu\rangle\\ &\quad=\frac{1}{\sqrt{N}}\sum_{\nu\in\mathbb{Z}/N\mathbb{Z}}\omega_N^{\nu\gamma}|\nu\log_g x,\nu\rangle,\end{aligned} \tag{37}$$

where we used identity (30).

(6) Measure this state in the computational basis. We obtain some pair $(\nu\log_g x,\nu)$ for a uniformly random $\nu\in\mathbb{Z}/N\mathbb{Z}$.

(7) Repeating the above gives a second pair $(\nu'\log_g x,\nu')$ with a uniformly random $\nu'\in\mathbb{Z}/N\mathbb{Z}$, independent of ν. With constant probability (at least $6/\pi^2\approx 0.61$), ν and ν' are coprime, in which case we can find integers λ and λ' such that $\lambda\nu+\lambda'\nu'=1$. Thus we can determine $\lambda\nu\log_g x+\lambda'\nu'\log_g x=\log_g x$.

This algorithm can be carried out for any cyclic group C, given a unique representation of its elements and the ability to efficiently compute products and inverses in C. To efficiently compute $f(\alpha,\beta)$, we must compute high powers of a group element, which can be done quickly by repeated squaring.

In particular, Shor's algorithm for computing discrete logarithms breaks the Diffie-Hellman key exchange protocol described above, in which $C=(\mathbb{Z}/p\mathbb{Z})^\times$. In Sec. IV.F we discuss further applications to cryptography, in which C is the group corresponding to an *elliptic curve*.

C. Hidden subgroup problem for finite Abelian groups

Algorithms 2 and 3 solve particular instances of a more general problem, the Abelian hidden subgroup problem (or Abelian HSP). We now describe this problem and show how it can be solved efficiently on a quantum computer.

Let G be a finite Abelian group with group operations written additively and consider a function $f:G\to S$,

[6] It is nevertheless an open question whether, given the ability to break the protocol, Eve can calculate discrete logarithms. Some partial results on this question are known (den Boer, 1990; Maurer and Wolf, 1999).

[7] Note that if we were to measure the ancilla register instead of discarding it, the outcome would be unhelpful: each possible value g^γ occurs with equal probability, and we cannot obtain γ from g^γ unless we know how to compute discrete logarithms.

where S is some finite set. We say that f *hides* the subgroup $H \leqslant G$ if

$$f(x) = f(y) \text{ if and only if } x - y \in H \tag{38}$$

for all $x, y \in G$. In the Abelian hidden subgroup problem, we are asked to find a generating set for H given the ability to query the function f.

It is clear that H can in principle be reconstructed from the entire truth table of f. Notice in particular that $f(0)=f(x)$ if and only if $x \in H$: the hiding function is constant on the hidden subgroup and does not take that value anywhere else. Furthermore, fixing any $y \in G$, we see that $f(y)=f(x)$ if and only if $x \in y+H := \{y+h : h \in H\}$, a *coset* of H in G with coset representative y. So f is constant on the cosets of H in G and distinct on different cosets.

The simplest example of the Abelian hidden subgroup problem is Simon's problem, in which $G=(\mathbb{Z}/2\mathbb{Z})^n$ and $H=\{0,x\}$ for some unknown $x \in (\mathbb{Z}/2\mathbb{Z})^n$. Simon's efficient quantum algorithm for this problem (Simon, 1997) led the way to Shor's algorithms for other instances of the Abelian HSP.

The period-finding problem discussed in Sec. IV.A is the Abelian HSP with $G=\mathbb{Z}/N\mathbb{Z}$. The subgroups of G are of the form $H=\{0,r,2r,\ldots,N-r\}$ (of order $|H| = N/r$), where r is a divisor of N. Thus a function hides H according to Eq. (38) precisely when it is r periodic, as in Eq. (25). We have already seen that such a subgroup can be found efficiently.

The quantum algorithm for computing discrete logarithms, as discussed in Sec IV.B, solves an Abelian hidden subgroup problem in the group $\mathbb{Z}/N\mathbb{Z} \times \mathbb{Z}/N\mathbb{Z}$. The function defined in Eq. (33) hides the subgroup

$$H = \{(\alpha, \alpha \log_g x) : \alpha \in \mathbb{Z}/N\mathbb{Z}\}. \tag{39}$$

Shor's algorithm computes $\log_g x$ by finding this hidden subgroup.

More generally, there is an efficient quantum algorithm to identify any hidden subgroup $H \leqslant G$ of a known finite Abelian group G. (In Sec. VII.C we relax the commutativity restriction to the requirement that H is a normal subgroup of G, which is always the case if G is Abelian.) The algorithm for the general Abelian hidden subgroup problem is as follows.

Algorithm 4 (*Abelian hidden subgroup problem*).

Input: A black-box function $f: G \to S$ hiding some $H \leqslant G$.

Problem: Find a generating set for H.

(1) Create a uniform superposition $|G\rangle$ over the elements of the group.

(2) Query the function f in an ancilla register, giving the state

$$\frac{1}{\sqrt{|G|}} \sum_{x \in G} |x, f(x)\rangle. \tag{40}$$

(3) Discard the ancilla register, giving the coset state

$$|s+H\rangle = \frac{1}{\sqrt{|H|}} \sum_{y \in H} |s+y\rangle \tag{41}$$

for some unknown uniformly random $s \in G$. Equivalently, the state can be described by the density matrix

$$\rho_H := \frac{1}{|G|} \sum_{s \in G} |s+H\rangle\langle s+H|. \tag{42}$$

(4) Apply the QFT over G to this state. According to the definition of the QFT in Eq. (9), the result is

$$\frac{1}{\sqrt{|H| \cdot |G|}} \sum_{\psi \in \hat{G}} \sum_{y \in H} \psi(s+y)|\psi\rangle = \sqrt{\frac{|H|}{|G|}} \sum_{\psi \in \hat{G}} \psi(s)\psi(H)|\psi\rangle, \tag{43}$$

where

$$\psi(H) := \frac{1}{|H|} \sum_{y \in H} \psi(y). \tag{44}$$

If $\psi(y)=1$ for all $y \in H$, then $\psi(H)=1$. On the other hand, if there is any $y \in H$ with $\psi(y) \neq 1$ (i.e., if the restriction of ψ to H is not the trivial character of H), then by the orthogonality of distinct irreducible characters (Theorem 6 in Appendix B) $\psi(H)=0$. Thus we have the state

$$\widehat{|s+H\rangle} := \sqrt{\frac{|H|}{|G|}} \sum_{\psi \in \hat{G}, \mathrm{Res}_H^G \psi = 1} \psi(s)|\psi\rangle \tag{45}$$

or, equivalently, the mixed quantum state

$$\hat{\rho}_H := \frac{|H|}{|G|} \sum_{\psi \in \hat{G}, \mathrm{Res}_H^G \psi = 1} |\psi\rangle\langle\psi|, \tag{46}$$

where $\mathrm{Res}_H^G \psi = 1$ means that $\psi(h)=1$ for all $h \in H$.

(5) Measure in the computational basis. We obtain one of the $|G|/|H|$ characters $\psi \in \hat{G}$ that is trivial on the hidden subgroup H, with every such character occurring with equal probability $|H|/|G|$. Letting $\ker \psi := \{g \in G : \psi(g)=1\}$ denote the kernel of the character ψ (which is a subgroup of G), we learn that $H \leqslant \ker \psi$.

(6) Repeat the entire process T times, obtaining characters $\psi_1, \ldots, \psi_T$, and output a generating set for K_T, where $K_t := \cap_{j=1}^t \ker \psi_j$. We are guaranteed that $H \leqslant K_t$ for any t. A simple calculation shows that if $K_t \neq H$, then $|K_{t+1}|/|K_t| \leqslant 1/2$ with probability at least $1/2$. Thus, we can choose $T=O(\log|G|)$ such that $K_T=H$ with high probability.

In summary, given a black-box function f hiding a subgroup H of a known finite Abelian group G, a quantum computer can determine H in time poly$(\log|G|)$ and, in particular, using only poly$(\log|G|)$ queries to the function f. Of course, this assumes that we can efficiently

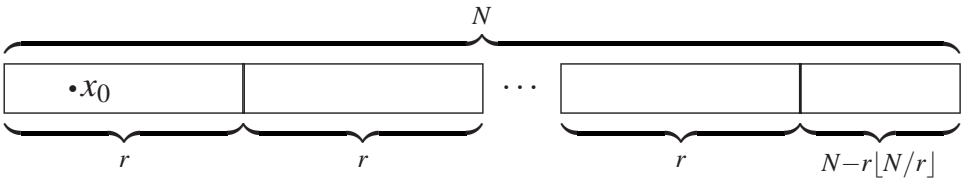


FIG. 2. Sampling a $\mathbb{Z}$-periodic function over $\mathbb{Z}/N\mathbb{Z}$.

implement group operations in G using some unique representation of its elements.

In contrast, the Abelian hidden subgroup problem is typically hard for classical computers. For example, an argument based on the birthday problem shows that even the simple case of Simon's problem [where $G=(\mathbb{Z}/2\mathbb{Z})^n$] has classical query complexity $\Omega(\sqrt{2^n})$ (Simon, 1997). While certain special cases are easy (for example, since the only subgroups of $\mathbb{Z}/p\mathbb{Z}$ with p prime are itself and the trivial subgroup, period finding over $\mathbb{Z}/p\mathbb{Z}$ is trivial) the classical query complexity of the Abelian HSP is usually exponential. In particular, one can show that if G has a set of N subgroups with trivial pairwise intersection, then the classical query complexity of the HSP in G is $\Omega(\sqrt{N})$. [For a proof in the case where $G=\mathbb{F}_q\times\mathbb{F}_q$, see de Beaudrap *et al.* (2002).]

D. Period finding over $\mathbb{Z}$

In the previous section, we showed that the Abelian HSP can be solved efficiently over any known finite Abelian group. In this section we consider the HSP over an infinite Abelian group, namely, $\mathbb{Z}$ (Shor, 1997). Similar ideas can be used to solve the HSP over any finitely generated Abelian group (Mosca and Ekert, 1999). (For an Abelian group that is not finitely generated, new ideas are required, as discussed in Sec. V.D.)

The HSP in $\mathbb{Z}$ is of interest when we are faced with a periodic function f over an unknown domain. For example, Shor's factoring algorithm (Sec. IV.E) works by finding the period of a function defined over $\mathbb{Z}$. Without knowing the factorization, it is unclear how to choose a finite domain whose size is a multiple of the unknown period, so we cannot immediately apply the period-finding algorithm from Sec. IV.A.

Of course, we cannot represent arbitrary integers on a computer with finitely many bits. Instead, we can restrict the function to the inputs $\{0,1,\ldots,N-1\}$ for some chosen N and perform Fourier sampling over $\mathbb{Z}/N\mathbb{Z}$. This can work even when the function is not precisely periodic over $\mathbb{Z}/N\mathbb{Z}$, provided N is sufficiently large. To simplify the implementation of the QFT, we can choose N to be a power of 2.

This approach can only work if the period is sufficiently small since otherwise we could miss the period entirely. We show how to choose N if given an *a priori* upper bound on the period. If we do not initially have such a bound, we can simply start with $N=2$ and repeatedly double N until the period-finding algorithm succeeds. The overhead incurred by this procedure is only $\mathrm{poly}(\log r)$.

Algorithm 5 (*Period finding over* $\mathbb{Z}$).

Input: A black box $f:\mathbb{Z}/N\mathbb{Z}\to S$ satisfying Eq. (25) for some $r\in\mathbb{Z}$ with $r^2<N$, where r does not necessarily divide N.

Problem: Determine r.

(1) Prepare the uniform superposition $|\mathbb{Z}/N\mathbb{Z}\rangle$.

(2) Query the function in an ancilla register, giving

$$\frac{1}{\sqrt{N}}\sum_{x\in\mathbb{Z}/N\mathbb{Z}}|x,f(x)\rangle. \tag{47}$$

(3) Discard the ancilla register, leaving the first register in a uniform superposition over those $x\in\mathbb{Z}/N\mathbb{Z}$ consistent with some particular function value. Since f is periodic with minimum period r, we obtain a superposition over points separated by r. The number of such points n depends on where the first point $x_0\in\{0,1,\ldots,r-1\}$ appears. When restricted to $\mathbb{Z}/N\mathbb{Z}$, the function has $\lfloor N/r\rfloor$ full periods and $N-r\lfloor N/r\rfloor$ remaining points, as shown in Fig. 2. Thus

$$n=\begin{cases}\lfloor N/r\rfloor+1 & \text{for } x_0<N-r\lfloor N/r\rfloor\\ \lfloor N/r\rfloor & \text{otherwise.}\end{cases} \tag{48}$$

In other words, we are left with the quantum state

$$\frac{1}{\sqrt{n}}\sum_{j=0}^{n-1}|x_0+jr\rangle, \tag{49}$$

where x_0 occurs nearly uniformly at random (specifically, it appears with probability n/N) and is unknown.

(4) Apply the Fourier transform over $\mathbb{Z}/N\mathbb{Z}$, giving

$$\frac{1}{\sqrt{nN}}\sum_{k\in\mathbb{Z}/N\mathbb{Z}}\omega_N^{kx_0}\sum_{j=0}^{n-1}\omega_N^{jkr}|k\rangle. \tag{50}$$

If we were lucky enough to choose a value of N for which $r|N$, then $n=N/r$ regardless of the value of x_0, and the sum over j gives $n\delta_{k \bmod n,0}$ by Eq. (30), so this state is identical to Eq. (31). But more generally, the sum over j in Eq. (50) is the geometric series

$$\sum_{j=0}^{n-1}\omega_N^{jkr}=\frac{\omega_N^{krn}-1}{\omega_N^{kr}-1}=\omega_N^{(n-1)kr/2}\frac{\sin(\pi krn/N)}{\sin(\pi kr/N)}. \tag{51}$$

(5) Measure in the computational basis. The probability of seeing a particular value k is

$$\Pr(k)=\frac{\sin^2(\pi krn/N)}{nN\sin^2(\pi kr/N)}. \tag{52}$$

From the case where $n=N/r$, we expect this distribution to be strongly peaked around values of k that are close to integer multiples of N/r. The probability of seeing k

$=\lfloor jN/r\rceil = jN/r+\varepsilon$ for some $j\in\mathbb{Z}$, where $\lfloor x\rceil$ denotes the nearest integer to x, is

$$\Pr(k=\lfloor jN/r\rceil) = \frac{\sin^2(\pi jn+\pi\varepsilon rn/N)}{nN\sin^2(\pi j+\pi\varepsilon r/N)} = \frac{\sin^2(\pi\varepsilon rn/N)}{nN\sin^2(\pi\varepsilon r/N)}. \quad (53)$$

Using the inequalities $4x^2/\pi^2 \leqslant \sin^2 x \leqslant x^2$ (where the lower bound holds for $|x|\leqslant \pi/2$ and can be applied since $|\varepsilon|\leqslant 1/2$), we find

$$\Pr(k=\lfloor jN/r\rceil) \geqslant \frac{4}{\pi^2 r}. \quad (54)$$

This bound shows that Fourier sampling produces a value of k that is the closest integer to one of the r integer multiples of N/r with probability $\Omega(1)$.

(6) To discover r given one of the values $\lfloor N/r\rceil$, divide by N to obtain a rational approximation to j/r that deviates by at most $1/2N$ and compute the positive integers a_i in the continued fraction expansion (CFE),

$$\frac{\lfloor jN/r\rceil}{N} = \cfrac{1}{a_1+\cfrac{1}{a_2+\cfrac{1}{a_3+\cdots}}}. \quad (55)$$

This expansion gives a sequence of successively better approximations to $\lfloor N/r\rceil$ by fractions, called the *convergents* of the CFE. Any fraction p/q with $|p/q-\lfloor jN/r\rceil/N|<1/2q^2$ will appear as one of the convergents [Hardy and Wright (1979), Theorem 184].

Since j/r differs by at most $1/2N$ from $\lfloor N/r\rceil$, the fraction j/r will appear as a convergent provided $r^2<N$. Thus, we carry out the CFE until we obtain the closest convergent to $\lfloor N/r\rceil$ whose denominator is smaller than our *a priori* upper bound on the period; this denominator must be equal to to r. These calculations can be done in polynomial time using standard techniques [see, e.g., Hardy and Wright (1979)].

Notice that period finding can efficiently determine the *order* of a given group element $g\in G$, the smallest $r\in\{1,2,\ldots\}$ such that $g^r=1$. This follows because the function $f:\mathbb{Z}\to G$ defined by $f(j)=g^j$ is periodic, with period equal to the order of g in G. In particular, this allows us to find the order of a cyclic group $C=\langle g\rangle$, as needed in Algorithm 3. In contrast, the classical query complexity of computing the order of a permutation of 2^n elements is $\Omega(2^{n/3}/\sqrt{n})$ (Cleve, 2004).

E. Factoring integers

Perhaps the best-known application of quantum computers is to the problem of factoring integers (Shor, 1997). At present, the mostly widely used public-key cryptosystem, RSA (Rivest *et al.*, 1978), is based on the presumed difficulty of this problem.[8] The fastest rigorously analyzed classical algorithm for factoring an integer N has running time $2^{O(\sqrt{\log N\log\log N})}$ [see, e.g., Pomerance (1987)] and the best known classical algorithm is believed to be the number field sieve (Buhler *et al.*, 1993), which is conjectured to run in time $2^{O((\log N)^{1/3}(\log\log N)^{2/3})}$. Both of these running times are superpolynomial in $\log N$. In contrast, a quantum computer can factor N in time $O(\log^3 N)$. Thus, the development of a large-scale quantum computer could have dramatic implications for the practice of cryptography.

We have already discussed the core of Shor's quantum factoring algorithm, the ability to perform period finding over the integers. It remains to see how factoring can be reduced to a particular instance of period finding.

To efficiently factor a given integer N, it suffices to efficiently produce some nontrivial factor of N (i.e., a factor other than 1 or N) with constant probability. The repeated use of such a subroutine, combined with an efficient primality testing algorithm (Miller, 1976; Rabin 1980; Agrawal *et al.*, 2004), can be used to find all the prime factors of N. It is easy to check whether 2 divides N, so we can focus on the case of N odd without loss of generality. Furthermore, it is straightforward to check whether N is a prime power or whether it is the kth power of any integer simply by computing $\sqrt[k]{N}$ for $k=2,3,\ldots,\log_2 N$, so we can assume that N has at least two distinct prime factors.

The reduction from finding some nontrivial factor of an odd N to order finding in the multiplicative group $(\mathbb{Z}/N\mathbb{Z})^\times$ is due to Miller (1976). Suppose we choose $a\in\{2,3,\ldots,N-1\}$ uniformly at random from those values that are coprime to N. Furthermore, assume that the order r of a is even. Then since $a^r=1 \bmod N$, we have $(a^{r/2})^2-1=0 \bmod N$ or, equivalently,

$$(a^{r/2}-1)(a^{r/2}+1)=0 \bmod N. \quad (56)$$

Since N divides the product $(a^{r/2}-1)(a^{r/2}+1)$, we might hope for $\gcd(a^{r/2}-1,N)$ to be a nontrivial factor of N. Notice that $\gcd(a^{r/2}-1,N)\neq N$ since if it were, the order of a would be at most $r/2$. Thus it suffices to ensure that $\gcd(a^{r/2}-1,N)\neq 1$, which holds if $a^{r/2}\neq -1 \bmod N$. In Lemma 2 below, we show that a random value of a satisfies these properties with probability at least $1/2$, provided N has at least two distinct prime factors. Thus the following quantum algorithm can be used to factor N.

Algorithm 6 (*Integer factorization*).

Input: An odd integer N with at least two distinct prime factors.

Problem: Determine some nontrivial factor of N.

(1) Choose a random $a\in\{2,3,\ldots,N-1\}$.

[8] The RSA protocol uses similar ideas to the Diffie-Hellman protocol (Sec. IV.B) but relies on a different assumption and achieves secure communication instead of key exchange. Note that breaking RSA might be easier than factoring. For elementary discussions of the details of RSA and related protocols, see (Menezes *et al.*, 1996; Buchmann, 2004).

(2) Compute $\gcd(a,N)$ using the Euclidean algorithm. If the result is different from 1, then it is a nontrivial factor of N, and we are done. More likely, $\gcd(a,N)=1$, and we continue.

(3) Using Algorithm 5, determine the order of $a \bmod N$. If r is odd, the algorithm has failed, and we return to step 6. If r is even, we continue.

(4) Compute $\gcd(a^{r/2}-1,N)$. If the result is different from 1, then it is a nontrivial factor of N. Otherwise, return to step 1.

Lemma 2. Suppose a is chosen uniformly at random from $(\mathbb{Z}/N\mathbb{Z})^\times$, where N is an odd integer with at least two distinct prime factors. Then with probability at least $1/2$, the multiplicative order r of $a \bmod N$ is even and $a^{r/2} \neq -1 \bmod N$.

Proof. Suppose $N=p_1^{m_1}\cdots p_k^{m_k}$ is the factorization of N into powers of $k \geqslant 2$ distinct odd primes. By the Chinese remainder theorem, there are unique values $a_i \in \mathbb{Z}/p_i^{m_i}\mathbb{Z}$ such that $a=a_i \bmod p_i^{m_i}$. Let r_i be the multiplicative order of $a_i \bmod p_i^{m_i}$ and let 2^{c_i} be the largest power of 2 that divides r_i. We claim that if r is odd or if $a^{r/2} = -1 \bmod N$, then $c_1=\cdots=c_k$. Since $r=\mathrm{lcm}(r_1,\ldots,r_k)$, we have $c_1=\cdots=c_k=0$ when r is odd. On the other hand, if r is even and $a^{r/2}=-1 \bmod N$, then for each i we have $a^{r/2}=-1 \bmod p_i^{m_i}$, so r_i does not divide $r/2$; but we know that r/r_i is an integer, so it must be odd, which implies that each r_i has the same number of powers of 2 in its prime factorization.

Now we claim that the probability of any given c_i taking on any particular value is at most $1/2$, which implies that $\Pr(c_1=c_2) \leqslant 1/2$, and the desired conclusion follows. To see this, consider a chosen uniformly at random from $(\mathbb{Z}/N\mathbb{Z})^\times$ or, equivalently, each a_i chosen uniformly at random from $(\mathbb{Z}/p_i^{m_i}\mathbb{Z})^\times$. The order of the latter group is $\varphi(p_i^{m_i})=(p_i-1)p_i^{m_i}=2^{d_i}q_i$ for some positive integer d_i and some odd integer q_i. The number of $a_i \in (\mathbb{Z}/p_i^{m_i}\mathbb{Z})^\times$ of odd order is q_i and the number of a_i's with any particular $c_i \in \{1,\ldots,d_i\}$ is $2^{c_i-1}q_i$. In particular, the highest-probability event is $c_i=d_i$, which happens with probability only $1/2$. ∎

F. Breaking elliptic curve cryptography

As discussed in Sec. IV.B, Shor's algorithm allows quantum computers to break cryptographic protocols based on the presumed hardness of the discrete logarithm problem in $(\mathbb{Z}/N\mathbb{Z})^\times$, such as the Diffie-Hellman key exchange protocol. However, Shor's algorithm works equally well for calculating discrete logarithms in any finite group, provided only that group elements can be represented uniquely and operated on efficiently. In particular, quantum computers can also efficiently calculate discrete logarithms over the group corresponding to an elliptic curve, thereby breaking elliptic curve cryptography.

An elliptic curve is a cubic, nonsingular, planar curve over some field. (The terminology has to do with a connection to elliptic functions.) For simplicity, suppose we choose a field with characteristic not equal to 2 or 3. (Cryptographic applications often use the field $\mathbb{F}_{2^n}$ of characteristic 2, but the definition of an elliptic curve is slightly more complicated in this case.) Then, by suitable linear transformations, any elliptic curve can be rewritten in the form of the Weierstraß equation

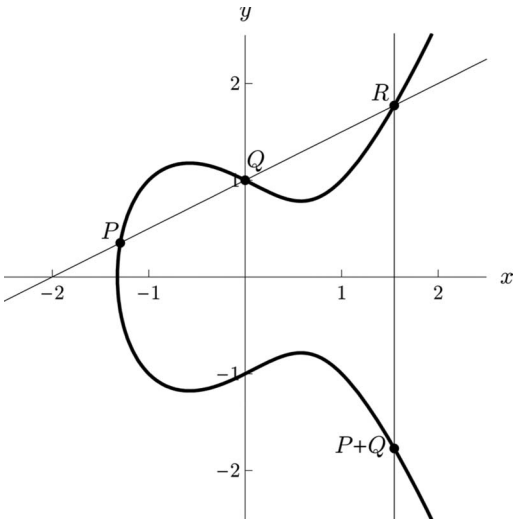


FIG. 3. The group law for an elliptic curve: $P+Q=-R$. The points P and Q sum to the point $-R$, where R is the intersection between the elliptic curve and the line through P and Q and $-R$ is obtained by the reflection of R about the x axis.

$$y^2 = x^3 + ax + b, \tag{57}$$

where a,b are parameters. The set of points (x,y) satisfying this equation forms an elliptic curve. To be nonsingular, the *discriminant* $\Delta := -16(4a^3+27b^2)$ must be nonzero. Typically, one considers elliptic curves in the projective plane $\mathbb{P}^2$ rather than the affine plane, which means that one point at infinity must be included in the set of solutions. (For further details on the concepts of projective curves, points at infinity, and nonsingularity, see Appendix C.)

An example of an elliptic curve over the field $\mathbb{R}$ (namely, the curve $y^2=x^3-x+1$) is shown in Fig. 3. Although such pictures are helpful for developing intuition about elliptic curves, it is useful in cryptographic applications to have a curve whose points can be represented exactly with a finite number of bits, so we use curves over finite fields. For simplicity, we only consider the field $\mathbb{F}_p$, where p is a prime larger than 3.

Example. Consider the curve

$$E = \{(x,y) \in \mathbb{F}_7^2 : y^2 = x^3 - x + 1\} \tag{58}$$

over $\mathbb{F}_7$. It has $4a^3+27b^2=2 \bmod 7$, so it is nonsingular. It is straightforward to check that the points on this curve are

$$E = \{O,(0,1),(0,6),(1,1),(1,6),(2,0),(3,2),\\ (3,5),(5,3),(5,4),(6,1),(6,6)\}, \tag{59}$$

where O denotes the point at infinity.

In general, the number of points on an elliptic curve depends on the parameters a and b. However, Hasse's theorem says that $|\|E|-(p+1)| \leq 2\sqrt{p}$, so for large p the number of points is close to p.

An elliptic curve can be used to define an Abelian group by designating one point of the curve as the additive identity. Here we use the common convention that O, the point at infinity, is this special element (although in principle, it is possible to let any point play this role). It remains to define a binary operation $+$ that maps a pair of points on the curve to a new point on the curve in a way that satisfies the group axioms. To motivate the definition, consider the case of the field $\mathbb{R}$. Given two points $P, Q \in E$, their sum $P+Q$ is defined geometrically as follows. First assume that neither point is O. Draw a line through the points P and Q (or, if $P=Q$ draw the tangent to the curve at P) and let R denote the third point of intersection, defined to be O if the line is vertical. Then $P+Q$ is the reflection of R about the x axis, where the reflection of O is O. If one of P or Q is O, we draw a vertical line through the other point, so that $P+O=P$ as desired. Since O is the additive identity, we define $O+O=O$. Reflection about the x axis corresponds to negation, so we can think of the rule as saying that the three points of intersection of a line with the curve sum to O, as shown in Fig. 3.

It can be shown that $(E,+)$ is an Abelian group, where the inverse of $P=(x,y)$ is $-P=(x,-y)$. From the geometric definition, it is clear that this group is Abelian (the line through P and Q does not depend on which point is chosen first) and closed (we always choose $P+Q$ to be some point on the curve). The only remaining group axiom to check is associativity: we must show that $(P+Q)+T=P+(Q+T)$.

To define the group operation for a general field, it is useful to have an algebraic description of elliptic curve point addition. Let $P=(x_P,y_P)$ and $Q=(x_Q,y_Q)$. Provided $x_P \neq x_Q$, the slope of the line through P and Q is

$$\lambda = (y_Q - y_P)/(x_Q - x_P). \tag{60}$$

Computing the intersection of this line with Eq. (57), we find

$$x_{P+Q} = \lambda^2 - x_P - x_Q, \tag{61}$$

$$y_{P+Q} = \lambda(x_P - x_{P+Q}) - y_P. \tag{62}$$

If $x_P=x_Q$, there are two possibilities for Q: either $Q=(x_Q,y_Q)=(x_P,y_P)=P$ or $Q=(x_Q,y_Q)=(x_P,-y_P)=-P$. If $Q=-P$, then $P+Q=O$. On the other hand, if $P=Q$ (i.e., if we are computing $2P$), then Eqs. (61) and (62) hold with λ replaced by the slope of the tangent to the curve at P, namely,

$$\lambda = (3x_P^2 + a)/2y_P \tag{63}$$

(unless $y_P=0$, in which case the slope is infinite, so $2P=O$).

While the geometric picture does not necessarily make sense for the case of a finite field, we can take its algebraic description as a definition of the group operation. It is again obvious that addition of points, defined by these algebraic expressions, is commutative and closed. Associativity of the group operation can be verified by a direct calculation. This shows that $(E,+)$ is indeed an Abelian group.

Suppose we fix an elliptic curve group $(E,+)$ and choose a point $g \in E$. Then we can consider the subgroup $\langle g \rangle$, which is possibly the entire group if it happens to be cyclic. Using exponentiation in this group (which is multiplication in our additive notation), we can define analogs of Diffie-Hellman key exchange and related cryptosystems such as ElGamal. The security of these cryptosystems then relies on the assumption that it is hard to compute discrete logarithms in $\langle g \rangle$.

In practice, there are many details to consider when choosing an elliptic curve group for cryptographic purposes (Menezes *et al.*, 1996; Buchmann, 2004). Algorithms are known for calculating discrete logarithms on "supersingular" and "anomolous" curves that run faster than algorithms for the general case, so such curves should be avoided. At the same time, g should be chosen to be a point of high order. Curves with the desired hardness properties can be found efficiently, and in the general case it is not known how to solve the discrete logarithm problem over an elliptic curve group classically any faster than by general methods (see Sec. IV.B), which run in time $O(\sqrt{p})$.

However, using Shor's algorithm, a quantum computer can solve the discrete logarithm problem for an elliptic curve group over $\mathbb{F}_p$ in time poly$(\log p)$. Points on the curve can be represented uniquely by their coordinates, with a special symbol used to denote O. Addition of points on the curve can be computed using Eqs. (61) and (62), which involve only elementary arithmetic operations in the field. The most complex of these operations is the calculation of modular inverses, which can easily be done using Euclid's algorithm. For more details on the implementation of Shor's algorithm over elliptic curves, see Proos and Zalka (2003); Kaye (2005); and Cheung *et al.* (2008).

Elliptic curve cryptosystems are commonly viewed as being more secure than RSA for a given key size since the best classical algorithms for factoring run faster than the best classical algorithms for calculating discrete logarithms in elliptic curve groups. Thus in practice, much smaller key sizes are used in elliptic curve cryptography than in factoring-based cryptography. Ironically, Shor's algorithm takes a comparable number of steps for both factoring and computing discrete logarithms,[9] so it could actually be easier for quantum computers to break

[9]Naively, computing the group operations for an elliptic curve using Eqs. (61) and (62) requires slightly more operations than performing ordinary integer multiplication. However, there are ways to improve the running time of Shor's algorithm for computing discrete logarithms over elliptic curve groups, at least in certain cases (Cheung *et al.*, 2008).

present-day elliptic curve cryptosystems than to break RSA.

One can also define an Abelian group corresponding to a hyperelliptic curve, a curve of the form $y^2=f(x)$ for some suitable polynomial f of degree higher than 3. These groups are also candidates for cryptographic applications [see, e.g., Koblitz (1998)]. In general, such a group is referred to as the Jacobian of the curve; it is no longer isomorphic to the curve itself in the nonelliptic case. The elements of a general Jacobian can be represented uniquely and added efficiently, so that Shor's algorithm can also efficiently compute discrete logarithms over the Jacobian of a hyperelliptic curve.

G. Decomposing Abelian and solvable groups

Recall from Sec. IV.D that Shor's period-finding algorithm can be used to compute the order of a cyclic group $C=\langle g\rangle$, given the ability to efficiently represent and multiply elements of the group. More generally, given a black-box representation of some group, it would be useful to have a way of identifying the structure of that group. For certain kinds of groups, such decompositions can be obtained efficiently by a quantum computer.

These algorithms operate in the framework of black-box groups (Babai and Szemerédi, 1984). In this framework, the elements of a group G are represented uniquely by strings of length poly($\log|G|$), and we are given a black box that can compute products or inverses in G as desired. Of course, any algorithm that works in the black-box setting also works when the group is represented explicitly, say as a matrix group or as some known group. Note that computing the order of G in the black-box group setting is hard even when G is promised to be Abelian (Babai and Szemerédi, 1984).

Suppose we are given a generating set for a finite Abelian black-box group. Recall that by the fundamental theorem of finite Abelian groups any such group can be decomposed as a direct product $G\cong\mathbb{Z}/p_1^{r_1}\mathbb{Z}\times\cdots\times\mathbb{Z}/p_k^{r_k}\mathbb{Z}$ of cyclic subgroups of prime power order. By combining the solution of the Abelian HSP with classical techniques from computational group theory, there is an efficient quantum algorithm for determining the structure of the group (i.e., the values $p_i^{r_i}$), and, furthermore, for obtaining generators for each of the cyclic factors (Mosca, 1999; Cheung and Mosca, 2001). Note that this provides an alternative approach to factoring an integer N: by decomposing the multiplicative group $(\mathbb{Z}/N\mathbb{Z})^\times$, we learn its size $\varphi(N)$, which is sufficient to determine the factors of N (Miller, 1976; Shoup, 2005).

More generally, a similar decomposition can be obtained for any solvable group (Watrous, 2001a). A finite group G is called solvable if there exist elements $g_1,\ldots,g_m\in G$ such that

$$\{1\}=H_0\trianglelefteq H_1\trianglelefteq\cdots\trianglelefteq H_m=G, \tag{64}$$

where $H_j:=\langle g_1,\ldots,g_j\rangle$ for each $j=0,1,\ldots,m$ and where the notation $H_j\trianglelefteq H_{j+1}$ indicates that H_j is a normal subgroup of H_{j+1}, i.e., that $xH_j=H_jx$ for every $x\in H_{j+1}$. (Equivalently, G is solvable if its derived series contains the trivial subgroup.) Every Abelian group is solvable, but the converse does not hold; for example, $S_3\cong D_3$ is non-Abelian but solvable. Given a generating set for a black-box solvable group, there is an efficient probabilistic classical algorithm to find $g_1,\ldots,g_m$ satisfying Eq. (64) for some $m=\mathrm{poly}(\log|G|)$ (Babai *et al.*, 1995). To compute the order of G, it suffices to compute the orders of the quotient groups H_j/H_{j-1} for $j=1,\ldots,m$, which are necessarily cyclic. We cannot directly compute the orders of these groups using Shor's algorithm since we do not have unique encodings of their elements. However, Watrous showed that if we are given the uniform superposition $|H_{j-1}\rangle$, we can (probabilistically) compute $|H_j/H_{j-1}|$ using a modified version of Shor's algorithm and also (probabilistically) prepare the state $|H_j\rangle$. By recursing this procedure along the normal series given by Eq. (64) (starting with enough copies of $|H_0\rangle$ and maintaining enough copies of the intermediate states $|H_j\rangle$ to handle the cases where the algorithm fails), a quantum computer can calculate $|G|$ in polynomial time. By straightforward reductions, this also gives efficient quantum algorithms for testing membership in solvable groups and for deciding whether a subgroup of a solvable group is normal. Similar ideas give a method for determining the structure of any Abelian factor group G/H, where $H\trianglelefteq G$ (Watrous, 2001a); see also Ivanyos *et al.* (2003) for related work.

H. Counting points on curves

Suppose we are given a polynomial $f\in\mathbb{F}_q[x_1,\ldots,x_n]$ in n variables over the finite field $\mathbb{F}_q$. The set $H_f:=\{x\in\mathbb{F}_q^n: f(x)=0\}$ of solutions to the equation $f(x)=0$ is called a *hypersurface.* Counting the number of solutions $|H_f|$ of this equation is a fundamental computational problem. More generally, given m polynomials $f_1,\ldots,f_m\in\mathbb{F}_q[x_1,\ldots,x_n]$, we may be interested in the number of solutions to the system of equations $f_1(x)=\cdots=f_m(x)=0$. The complexity of such counting problems can be characterized in terms of at least five parameters: the number m of polynomials, the number n of variables, the degrees $\deg(f_i)$ of the polynomials, the size q of the finite field $\mathbb{F}_q$, and the characteristic p of the field, where $q=p^r$ and p is prime.

The complexity class #P characterizes the difficulty of counting the number of values x such that $f(x)=0$, where f is an efficiently computable function. One can show that for quadratic polynomials over $\mathbb{F}_2$, with no restrictions on the number n of variables and the number m of polynomials, the corresponding counting problem is #P-complete. As #P problems are at least as hard as NP problems (see Sec. II.D), we do not expect quantum computers to solve such counting problems in time poly(n,m). In fact, the counting problem is #P-hard even for a single polynomial in two variables (von zur Gathen *et al.*, 1997) provided we use a sparse representation that only lists the nonzero coefficients of the polynomial, which allows its degree to be exponential in the size of

its representation. Using a nonsparse representation, so that we aim for a running time polynomial in the degree, the computational complexity of such counting problems is a more subtle issue.

Here we are concerned with the counting problem for planar curves, meaning that we have $m=1$ polynomial in $n=2$ variables. (Appendix C contains some background information about curves over finite fields for readers unfamiliar with this topic.) A key parameter characterizing the complexity of this counting problem is the *genus* g of the curve. For a nonsingular, projective, planar curve f, the genus is $g=1/2(d-1)(d-2)$, where $d=\deg(f)$.

Schoof (1985) gave an algorithm to count the number of points on an elliptic curve (for which $g=1$) over $\mathbb{F}_q$ in time poly($\log q$). Following results by Pila (1990), Adleman and Huang (2001) generalized this result to hyperelliptic curves, giving an algorithm with running time $(\log q)^{O(g^2 \log g)}$, where g is the genus of the curve. For fields $\mathbb{F}_{p^r}$ with characteristic p, Lauder and Wan (2008) showed the existence of a deterministic algorithm for counting points with time complexity poly($p,r,\deg f$). While the former algorithm is efficient for $g=O(1)$ and the latter is efficient for $p=\text{poly}(\log q)$, neither is efficient without some restriction on the genus or the field characteristic.

On the other hand, Kedlaya (2006) explained how the quantum algorithm for determining the structure of an unknown finite Abelian group (Sec. IV.G) can be used to count the number of points on a planar curve of genus g over $\mathbb{F}_q$ in time poly($g,\log q$). It is probably fair to say that this constitutes not so much a new quantum algorithm but rather a novel application of known quantum algorithms to algebraic geometry.

In brief, Kedlaya's algorithm counts the solutions of a smooth projective curve C_f of genus g by determining the $2g$ nontrivial roots of the corresponding zeta function $Z_f(T)$, which are determined from the orders of the class groups $\text{Cl}_s(C_f)$ over the different base fields $\mathbb{F}_{p^s}$ for $s=1,\ldots,16g$. As the class groups are all finite Abelian groups, $|\text{Cl}_s(C_f)|$ can be computed in time poly($g,s,\log p$) by a quantum computer, thus giving an efficient quantum algorithm for the point counting problem. We explain some of the details below. For further information, see Hulek (2003); Lorenzini (1996); and the original article by Kedlaya.

The zeta function of a curve. Let the polynomial $f \in \mathbb{F}_p[X,Y]$ define a smooth, planar, projective curve C_f. To count the number of points on this curve in the projective plane $\mathbb{P}^2(\mathbb{F}_p)$, it is useful to consider extensions of the base field. For any positive integer r, we define

$$N_r := |C_f(\mathbb{F}_{p^r})|, \tag{65}$$

where

$$C_f(\mathbb{F}_{p^r}) := \{x \in \mathbb{P}^2(\mathbb{F}_{p^r}) : f(x) = 0\} \tag{66}$$

denotes the projective curve defined by f when viewed as a polynomial over $\mathbb{F}_{p^r}$.

In terms of these values, we can define the zeta function $Z_f(T)$ of the curve C_f, namely,

$$Z_f(T) := \exp\left(\sum_{r=1}^{\infty} \frac{N_r}{r} T^r\right), \tag{67}$$

with T a formal variable and the exponential function defined by the Taylor series $\exp(x)=\Sigma_{j=0}^{\infty} x^j/j!$. Whereas the Riemann zeta function is used to study the elements and primes of the ring $\mathbb{Z}$, the zeta function of a curve captures the ideals and prime ideals of the ring $\mathbb{F}_p[X,Y]/(f)$, where (f) denotes the ideal generated by f.

From the proof of Weil's Riemann hypothesis for curves [see, e.g., Lorenzini (1996)], the zeta function of a smooth projective curve C_f of genus g has the form

$$Z_f(T) = \frac{Q_f(T)}{(1-pT)(1-T)}, \tag{68}$$

where $Q_f(T)$ is a polynomial of degree $2g$ with integer coefficients. Moreover, $Q_f(T)$ has the factorization

$$Q_f(T) = \prod_{j=1}^{2g} (1-\alpha_j T) \tag{69}$$

with $\alpha_{g+j}=\alpha_j^*$ and $|\alpha_j|=\sqrt{p}$ for all j. By considering the rth derivative of $Z_f(T)$ at $T=0$, it is easy to see that the values α_j determine the numbers $N_1,N_2,\ldots$, and in particular

$$N_r = p^r + 1 - \sum_{j=1}^{2g} \alpha_j^r \tag{70}$$

for all r. Thus, if we know the integer coefficients of the degree $2g$ polynomial $Q_f(T)$, we can infer the number of points on the curve $f(x)=0$ over $\mathbb{P}^2(\mathbb{F}_{p^r})$. Kedlaya's algorithm calculates $Z_f(T)$ and hence $Q_f(T)$ by relating it to the *class group* of the curve, a finite Abelian group.

The class group of a function field. A *divisor* D on a curve C over $\mathbb{F}_p$ is a finite formal sum over points on the curve extended to the algebraic closure $\bar{\mathbb{F}}_p$ of $\mathbb{F}_p$, namely,

$$D = \sum_{P \in C(\bar{\mathbb{F}}_p)} c_P \cdot P. \tag{71}$$

To be a divisor, D must satisfy three conditions: (1) $c_P \in \mathbb{Z}$ for all P, (2) $\Sigma_P |c_P|$ is finite, and (3) D is invariant under the Frobenius automorphism $\phi: x \mapsto x^p$, i.e., $c_P = c_{\phi(P)}$ for all P. The degree of D is the integer $\deg(D) = \Sigma_P c_P$. Under pointwise addition of the coefficients c_P, the divisors of degree 0 form the group

$$\text{Div}(C) := \{D : \deg(D) = 0\}. \tag{72}$$

As explained in Appendix C, for any curve C_f one can define the function field $\mathbb{F}_p(C_f)$, the field of rational functions $\{g=g_1/g_2 : g_2 \neq 0\}$, where g_1 and g_2 are homogeneous polynomials of equal degree modulo (f), such that g is a function on the projective curve $C_f(\bar{\mathbb{F}}_p)$. For each such nonzero rational function $g \in \mathbb{F}_p^{\times}(C_f)$ we define the corresponding principal divisor

$$\begin{aligned}\mathrm{div}(g) &:= \sum_{P\in C_f(\bar{\mathbb{F}}_p)} \mathrm{ord}_P(g)\cdot P \\ &= \sum_{P\in C_f(\bar{\mathbb{F}}_p)} \mathrm{ord}_P(g_1)\cdot P - \sum_{P\in C_f(\bar{\mathbb{F}}_p)} \mathrm{ord}_P(g_2)\cdot P,\end{aligned} \tag{73}$$

where the non-negative integer $\mathrm{ord}_P(g_i)$ is the multiplicity of P as a solution to $g_i=0$. In particular, $\mathrm{ord}_P(g_i)\geq 1$ if and only if $g_i(P)=0$ and $\mathrm{ord}_P(g_i)=0$ when $g_i(P)\neq 0$.

For each principal divisor we have $\deg[\mathrm{div}(g)]=0$. For a rational curve such as the straight line $C=\mathbb{P}^1$, the converse holds as well: the only divisors of degree 0 are the principal divisors of the curve. But it is an important fact that for general curves the converse does not hold. For curves that are not rational, i.e., curves of positive genus such as elliptic curves, the class group captures the relationship between the group $\mathrm{Div}(C_f)$ and its subgroup of principal divisors.

There is a crucial equivalence relation $\sim$ among divisors defined by

$$\begin{aligned}&D_1 \sim D_2 \text{ if and only if } D_1 - D_2 \\ &\qquad \text{is a principal divisor.}\end{aligned} \tag{74}$$

Finally, the (divisor) class group $\mathrm{Cl}(C)$ of curve C is defined as the group of degree 0 divisors modulo this equivalence relation,

$$\mathrm{Cl}(C) := \mathrm{Div}(C)/\sim . \tag{75}$$

Returning to the theory of zeta functions, it is known that the order of $\mathrm{Cl}(C_f)$ can be expressed in terms of the roots α_j of $Z_f(T)$ as

$$|\mathrm{Cl}(C_f)| = \prod_{j=1}^{2g} (1-\alpha_j). \tag{76}$$

This fact establishes a close connection between the number of points $N_1=|C_f(\mathbb{F}_p)|$ on a curve and the size $|\mathrm{Cl}(C_f)|$ of its class group.

All of the above can repeated while interpreting the polynomial f as an element of the extended ring $\mathbb{F}_{p^s}[X,Y]$. Indicating this change of the base field from $\mathbb{F}_p$ to its degree s extension $\mathbb{F}_{p^s}$ with a parenthesized superscript, the zeta function $Z_f^{(s)}(T)$ has $2g$ nontrivial roots $\alpha_j^{(s)}=\alpha_j^s$ for $j=1,\ldots,2g$ and the class group $\mathrm{Cl}^{(s)}(C_f)$ has order

$$|\mathrm{Cl}^{(s)}(C_f)| = \prod_{j=1}^{2g} (1-\alpha_j^s). \tag{77}$$

Observe that the change of base field affects the class group since the new divisors must be invariant under the Frobenius automorphism $\phi^s: x\mapsto x^{p^s}$ [which is a weaker restriction than the corresponding condition over $\mathbb{F}_p$, making $\mathrm{Div}^{(s)}(C_f)$ larger than $\mathrm{Div}(C_f)$], while the group of principal divisors now allows all rational functions $g\in\mathbb{F}_{p^s}(C_f)$.

To illustrate the above definitions, we present the following example of the class group of an elliptic curve.

Example (Point counting and the class group of an elliptic curve). Consider the elliptic curve E over $\mathbb{F}_2$ defined by the equation $Y^2+XY+X^3+1=0$. The projective version of E is defined by the homogeneous equation $Y^2Z+XYZ+X^3+Z^3=0$. We want to consider the number of points N_r in the projective space $\mathbb{P}^2(\mathbb{F}_{2^r})$ for various r.

It is not hard to see that $N_1=4$ with the following four solutions:

	P_0	P_1	P_2	P_3
$(X:Y:Z)$	(0:1:0)	(1:0:1)	(0:1:1)	(1:1:1)

For the first extension field, there are $N_2=8$ elements in $E(\mathbb{F}_4)$: in addition to the previous four points, we now also have the solutions

	P_4	P_5	P_6	P_7
$(X:Y:Z)$	$(\omega:0:1)$	$(\omega:\omega:1)$	$(\omega^2:0:1)$	$(\omega^2:\omega^2:1)$

with ω an element of the field $\mathbb{F}_4$ satisfying $\omega^2=\omega+1$.

In general, it can be shown that the number of points on $E(\mathbb{F}_{2^r})$ is

$$N_r = 2^r + 1^r - \alpha^r - \bar{\alpha}^r \tag{78}$$

for any r, where $\alpha := -\frac{1}{2}+\frac{1}{2}\sqrt{-7}$.

To explore the class group $\mathrm{Cl}(E)$ of this curve, we start by considering some principal divisors. For the linear functions in X,Y,Z we find the following (degree 3) divisors:

$\mathrm{ord}_P(f)$	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7
X	1		2					
Y		1			1		1	
Z	3							
$X+Y$				1		1		1
$X+Z$	1	1		1				
$Y+Z$			1	2				
$X+Y+Z$		2	1					

From this table we see, for example, that the principal divisor of X/Z equals $-2P_0+2P_2$ and that $\mathrm{div}[(X+Y+Z)/(X+Z)]=-P_0+P_1+P_2-P_3$. [Note also that in this function field we have equalities such as $X^2+YZ=(X+Z)^2(Y+Z)/(X+Y+Z)$, which confirms that $\mathrm{div}(X^2+YZ)=2\,\mathrm{div}(X+Z)+\mathrm{div}(Y+Z)-\mathrm{div}(X+Y+Z)=2P_0+4P_3$.]

One can also show that P_0-P_1 is not a principal divisor and hence that $\mathrm{Cl}(E)$ is nontrivial. In fact, there are four different elements C_j of the class group, which we can indicate by the representatives

C_0	C_1	C_2	C_3
0	P_0-P_1	P_0-P_2	P_0-P_3

(Note, however, that these representatives are far from unique as, for example, $0\sim -P_0+P_1+P_2-P_3=\mathrm{div}[(X+Y+Z)/(X+Z)]$.) One can verify that the elements of $\mathrm{Cl}(E)$ act as the group $\mathbb{Z}/4\mathbb{Z}$, with $C_x+C_y\sim C_{x+y}$ for any $x,y\in\mathbb{Z}/4\mathbb{Z}$.

Performing similar calculations over the extension field $\mathbb{F}_{2^s}$, one can show that in general

$$|\mathrm{Cl}^{(s)}(E)| = (1-\alpha^s)(1-\bar{\alpha}^s) = 2^s + 1 - \alpha^s - \bar{\alpha}^s, \tag{79}$$

where α is as in Eq. (78). This concludes our example.

While for elliptic curves the number of points on the curve equals the number of elements of the corresponding class group, this coincidence does not persist for general curves with genus different from 1. However, the class group is nevertheless always a finite Abelian group, which can be explored using the quantum algorithm of Sec. IV.G.

Kedlaya's algorithm. Finally, we describe the quantum algorithm of Kedlaya (2006) for counting the points on a curve over a finite field.

Algorithm 7 (Point counting).

Input: A nonsingular, planar, projective curve C_f defined by a polynomial $f \in \mathbb{F}_q[X,Y]$.

Problem: Determine the number of solutions $|C_f(\mathbb{F}_{q^r})|$ of the equation $f=0$ in the projective plane $\mathbb{P}^2(\mathbb{F}_{q^r})$.

(1) Let $g=\frac{1}{2}(d-1)(d-2)$ be the genus of the curve, where $d=\deg(f)$.

(2) For $s=1,2,\ldots,16g$, (a) construct the class group $\mathrm{Cl}^{(s)}(C_f)$ and (b) using the algorithm of Sec. IV.G, determine $|\mathrm{Cl}^{(s)}(C_f)|$.

(3) Using the calculated group sizes and the equalities

$$|\mathrm{Cl}^{(s)}(C_f)| = \prod_{j=1}^{2g} (1-\alpha_j^s) \tag{80}$$

for $s=1,2,\ldots,16g$, determine the roots α_j.

(4) Compute $N_r=|C_f(\mathbb{F}_{q^r})|=q^r+1-\Sigma_{j=1}^{2g}\alpha_j^r$.

Several aspects of this algorithm are beyond the scope of this article, most notably the issue of uniquely representing and manipulating the elements of the class group $\mathrm{Cl}(C_f)$ in such a way that they can be sampled (nearly) uniformly, facilitating finding a set of generators. For an explanation of this and other issues, see the original article by Kedlaya, and references therein.

In conclusion, the above quantum algorithm has running time polynomial in the parameters $\log p^r$ and g, whereas the best known classical algorithms are either exponential in g (Adleman and Huang, 2001) or exponential in $\log p$ (Lauder and Wan, 2008). Whether it is possible to generalize Kedlaya's algorithm for curves to more general surfaces, i.e., to polynomials f with more than two variables, remains an open question. The best known classical result for this problem is that of Lauder and Wan (2008), who described an algorithm with running time $\mathrm{poly}(p^n, r^n, \deg(f)^{n^2})$.

V. QUANTUM ALGORITHMS FOR NUMBER FIELDS

A. Pell's equation

Given a squarefree integer d (i.e., an integer not divisible by any perfect square), the Diophantine equation

TABLE I. Some examples of fundamental solutions of Pell's equation $x^2-dy^2=1$ for different input values d (Jozsa, 2003).

d	x_1	y_1
2	3	2
3	2	1
5	9	4
⋮	⋮	⋮
13	649	180
14	15	4
⋮	⋮	⋮
6009	1316340106327253158	16981146611578034 51
	92594469510599473 88	68894923788314657 66
	$4013975 \approx 1.3\times 10^{44}$	$81644 \approx 1.6\times 10^{42}$
6013	40929908599	527831340
⋮	⋮	⋮

$$x^2 - dy^2 = 1 \tag{81}$$

is known as Pell's equation. This appellation provides an example of Stigler's law of eponymy (Stigler, 1980) in action, as Pell had nothing whatsoever to do with the equation. The misattribution is apparently due to Euler, who confused Pell with a contemporary, Brouncker, who had actually worked on the equation. In fact, Pell's equation was studied in ancient India, where (inefficient) methods for solving it were developed hundreds of years before Pell (Lenstra, 2002). (Indeed, Lenstra has suggested that most likely Pell was named after the equation.)

The left hand side of Pell's equation can be factored as

$$x^2 - dy^2 = (x + y\sqrt{d})(x - y\sqrt{d}). \tag{82}$$

Note that a solution of the equation $(x,y) \in \mathbb{Z}^2$ can be encoded uniquely as the real number $x+y\sqrt{d}$: since $\sqrt{d}$ is irrational, $x+y\sqrt{d}=w+z\sqrt{d}$ if and only if $(x,y)=(w,z)$. Thus we can also refer to the number $x+y\sqrt{d}$ as a solution of Pell's equation.

There is clearly no loss of generality in restricting our attention to *positive* solutions of the equation, namely, those for which $x>0$ and $y>0$. It is straightforward to show that if $x_1+y_1\sqrt{d}$ is a positive solution, then $(x_1+y_1\sqrt{d})^n$ is also a positive solution for any $n \in \mathbb{N}$. In fact, with $x_1+y_1\sqrt{d}$ the smallest positive solution of the equation, called the *fundamental solution,* one can show that all positive solutions equal $(x_1+y_1\sqrt{d})^n$ for some $n \in \mathbb{N}$. Thus, even though Pell's equation has an infinite number of solutions, we can in a sense find them all by finding the fundamental solution.

Some examples of fundamental solutions for various values of d are shown in Table I. Notice that while the size of the fundamental solution generally increases with increasing d, the behavior is far from monotonic: for example, x_1 has 44 decimal digits when $d=6009$, but only 11 decimal digits when $d=6013$. In general, though, it is possible for the solutions to be very large: the size of

$x_1+y_1\sqrt{d}$ is only upper bounded by $2^{O(\sqrt{d}\log d)}$. Thus it is not even possible to write down the fundamental solution with poly$(\log d)$ bits.

To get around this difficulty, we define the *regulator* of the fundamental solution,

$$R := \log(x_1 + y_1\sqrt{d}). \tag{83}$$

Since $R=O(\sqrt{d}\log d)$, we can write down $\lfloor R\rceil$, the nearest integer to R, using $O(\log d)$ bits. Since R is an irrational number, determining only its integer part may seem unsatisfactory, but in fact given $\lfloor R\rceil$ there is a classical algorithm to compute n digits of R in time poly$(\log d,n)$. Thus we will be satisfied with an algorithm that finds the integer part of R in time poly$(\log d)$. The best known classical algorithm for this problem runs in superpolynomial time (for more details, see Sec. V.E). In contrast, Hallgren (2007) gave a polynomial-time quantum algorithm for computing $\lfloor R\rceil$. For a self-contained review of Hallgren's algorithm, see Jozsa (2003).

B. From Pell's equation to the unit group

Given a squarefree positive integer d, the quadratic number field $\mathbb{Q}[\sqrt{d}]$ is defined as

$$\mathbb{Q}[\sqrt{d}] := \{x + y\sqrt{d} : x,y \in \mathbb{Q}\}. \tag{84}$$

It is easy to check that $\mathbb{Q}[\sqrt{d}]$ is a field with the usual addition and multiplication operations. We also define an operation called *conjugation* as

$$\overline{x + y\sqrt{d}} := x - y\sqrt{d}. \tag{85}$$

One can easily check that conjugation of elements of $\mathbb{Q}[\sqrt{d}]$ has many of the same properties as complex conjugation, and indeed $\mathbb{Q}[\sqrt{d}]$ behaves in many respects like $\mathbb{C}$, with $\sqrt{d}$ taking the place of the imaginary unit $i=\sqrt{-1}$. Defining the ring $\mathbb{Z}[\sqrt{d}]\subset\mathbb{Q}[\sqrt{d}]$ as

$$\mathbb{Z}[\sqrt{d}] := \{x + y\sqrt{d} : x,y \in \mathbb{Z}\}, \tag{86}$$

we see that solutions of Pell's equation correspond to those $\xi\in\mathbb{Z}[\sqrt{d}]$ satisfying $\xi\bar{\xi}=1$.

Notice that any solution of Pell's equation, $\xi\in\mathbb{Z}[\sqrt{d}]$, has the property that its multiplicative inverse over $\mathbb{Q}[\sqrt{d}]$, $\xi^{-1}=\bar{\xi}/\xi\bar{\xi}=\bar{\xi}$, is also an element of $\mathbb{Z}[\sqrt{d}]$. In general, an element of a ring with an inverse that is also an element of the ring is called a unit. In $\mathbb{Z}$, the only units are ±1, but in other rings it is possible to have more units.

It should not be a surprise that the units of $\mathbb{Z}[\sqrt{d}]$ are closely related to the solutions of Pell's equation. In particular, $\xi=x+y\sqrt{d}$ is a unit in $\mathbb{Z}[\sqrt{d}]$ if and only if $\xi\bar{\xi}=x^2-dy^2=\pm1$. To see this, we note that

$$\xi^{-1} = \bar{\xi}/\xi\bar{\xi} = (x - y\sqrt{d})/(x^2 - dy^2) \tag{87}$$

and if $x^2-dy^2=\pm1$, then $\xi^{-1}=\pm\bar{\xi}\in\mathbb{Z}[\sqrt{d}]$. Conversely, if $\xi^{-1}\in\mathbb{Z}[\sqrt{d}]$, then so is

$$\xi^{-1}\overline{\xi^{-1}} = \frac{(x - y\sqrt{d})(x + y\sqrt{d})}{(x^2 - dy^2)^2} = \frac{1}{x^2 - dy^2}, \tag{88}$$

which shows that $x^2-dy^2=\pm1$.

The set of units in $\mathbb{Z}[\sqrt{d}]$ forms a group under multiplication called the unit group. This group is given by $\{\pm\epsilon_1^n : n\in\mathbb{Z}\}$, where ϵ_1 is the fundamental unit, the smallest unit greater than 1. The proof of this fact is essentially the same as the proof that all solutions of Pell's equation are powers of the fundamental solution.

If we can find ϵ_1, then it is straightforward to find all the solutions of Pell's equation. If $\epsilon_1=x+y\sqrt{d}$ has $x^2-dy^2=+1$, then the units are precisely the solutions of Pell's equation. On the other hand, if $x^2-dy^2=-1$, then $\epsilon_2 := \epsilon_1^2$ satisfies $\epsilon_2\bar{\epsilon}_2=\epsilon_1^2\bar{\epsilon}_1^2=(-1)^2=1$; in this case the solutions of Pell's equation are $\{\pm\epsilon_1^{2n} : n\in\mathbb{Z}\}$. Thus our goal is to find ϵ_1. Just as in our discussion of the solutions to Pell's equation, ϵ_1 is too large to write down, so instead we compute the regulator of the fundamental unit, $\mathcal{R} := \log\epsilon_1$.

Example. Consider the quadratic number field $\mathbb{Q}[\sqrt{5}]$ and the corresponding ring $\mathbb{Z}[\sqrt{5}]$. The unit group of $\mathbb{Z}[\sqrt{5}]$ has the fundamental unit $\epsilon_1=2+\sqrt{5}$, whose regulator is $\mathcal{R}=\log(2+\sqrt{5})\approx1.44$. Here $\epsilon_1\bar{\epsilon}_1=-1$, so the fundamental solution of Pell's equation is $x_1+y_1\sqrt{5}=\epsilon_1^2=9+4\sqrt{5}$. Thus the set of positive solutions to Pell's equation $x^2-5y^2=1$ is

$$\{(x_k,y_k) : x_k + y_k\sqrt{5} = (9 + 4\sqrt{5})^k, k \in \mathbb{N}\}. \tag{89}$$

C. Periodic function for Pell's equation

To define a periodic function that encodes $\mathcal{R}$, we need to introduce the concept of an ideal of a ring (and more specifically a *principal ideal*). For any ring R, we say that $I\subseteq R$ is an ideal if it is closed under integer linear combinations and under multiplication by arbitrary elements of R. For example, $2\mathbb{Z}$ is an ideal of $\mathbb{Z}$. We say that an ideal is principal if it is generated by a single element of the ring, i.e., if it is of the form αR for some $\alpha\in R$; thus $2\mathbb{Z}$ is a principal ideal.

Principal ideals are useful because the function mapping the ring element $\xi\in\mathbb{Z}[\sqrt{d}]$ to the principal ideal ξR is periodic and its periodicity corresponds to the units of $\mathbb{Z}[\sqrt{d}]$. Specifically, $\xi\mathbb{Z}[\sqrt{d}]=\zeta\mathbb{Z}[\sqrt{d}]$ if and only if $\xi=\zeta\epsilon$, where ϵ is a unit in $\mathbb{Z}[\sqrt{d}]$. To see this, note that if ϵ is a unit, then $\xi\mathbb{Z}[\sqrt{d}]=\zeta\epsilon\mathbb{Z}[\sqrt{d}]=\zeta\mathbb{Z}[\sqrt{d}]$ since $\epsilon\mathbb{Z}[\sqrt{d}]=\mathbb{Z}[\sqrt{d}]$ by the definition of a unit. Conversely, suppose that $\xi\mathbb{Z}[\sqrt{d}]=\zeta\mathbb{Z}[\sqrt{d}]$; then, since $1\in\mathbb{Z}[\sqrt{d}]$, we have $\xi\in\xi\mathbb{Z}[\sqrt{d}]=\zeta\mathbb{Z}[\sqrt{d}]$, so there is some $\mu\in\mathbb{Z}[\sqrt{d}]$ satisfying $\xi=\zeta\mu$. Similarly, $\zeta\in\zeta\mathbb{Z}[\sqrt{d}]=\xi\mathbb{Z}[\sqrt{d}]$, so there is some $\nu\in\mathbb{Z}[\sqrt{d}]$ satisfying $\zeta=\xi\nu$. Thus we have $\xi=\zeta\mu=\xi\nu\mu$. This shows that $\nu\mu=1$, so μ and ν are units (indeed, $\nu=\mu^{-1}$).

As a result, the function $g(\xi)=\xi\mathbb{Z}[\sqrt{d}]$ is (multiplicatively) periodic with period ϵ_1. In other words, letting $\xi=e^z$, the function

$$h(z) = e^z \mathbb{Z}[\sqrt{d}] \tag{90}$$

is (additively) periodic with period $\mathcal{R}$. However, we cannot simply use this function since it is not possible to succinctly represent the values it takes.

To define a more suitable periodic function, one can use the concept of a *reduced* ideal. We will not describe the details here. However, one can show that there are only finitely many reduced principal ideals and indeed only $O(d)$ of them, so that we can represent a reduced principal ideal using poly(log d) bits.

It is also helpful to have a way of measuring the distance δ of any principal ideal from the *unit ideal*, $1\mathbb{Z}[\sqrt{d}] = \mathbb{Z}[\sqrt{d}]$. Such a function can be defined by

$$\delta(\xi\mathbb{Z}[\sqrt{d}]) := \log|\xi/\bar{\xi}| \bmod \mathcal{R}. \tag{91}$$

Notice that the unit ideal has distance $\delta(1\mathbb{Z}[\sqrt{d}]) = \log|1/1| \bmod \mathcal{R} = 0$, as desired. Furthermore, the distance function does not depend on which generator we choose to represent an ideal since two equivalent ideals have generators that differ by some unit $\epsilon = \epsilon_1^n$ and

$$\delta(\varepsilon\mathbb{Z}[\sqrt{d}]) = 2\log|\varepsilon| \bmod \mathcal{R} = 0. \tag{92}$$

With this definition of distance, one can show that there is a reduced ideal close to any nonreduced ideal.

The periodic function $f(z)$ used in Hallgren's algorithm is defined as the reduced principal ideal whose distance from the unit ideal is maximal among all reduced principal ideals of distance at most z (together with the distance from z to ensure that the function is injective within each period). In other words, we select the reduced principal ideal "to the left of or at z."

This function f is periodic with period $\mathcal{R}$, and one can show that it can be computed in time poly(log d). However, since $\mathcal{R}$ is in general irrational, it remains to see how to perform period finding for such a function.

D. Period finding over $\mathbb{R}$

Suppose we are given a function $f:\mathbb{R}\to S$ satisfying

$$f(x) = f(y) \text{ if and only if } (x-y)/r \in \mathbb{Z} \tag{93}$$

for some $r \in \mathbb{R}$ for all $x, y \in \mathbb{R}$. Here we consider how Shor's period-finding algorithm (Sec. IV.D) can be adapted to find an approximation to r even if it happens to be irrational (Hallgren, 2007).

Of course, to perform period finding on a digital computer, we must discretize the function. We must be careful about how we perform this discretization. For example, suppose that $S=\mathbb{R}$. If we simply evaluate f at equally spaced points and round the resulting values to obtain integers, there is no reason for the function values corresponding to inputs separated by an amount close to the period to be related in any way whatsoever. It could be that the discretized function is injective, carrying absolutely no information about the period.

Instead we discretize in such a way that the resulting function is *pseudoperiodic*. We say that $f:\mathbb{Z}\to S$ is pseudoperiodic at $k \in \mathbb{Z}$ with period $r \in \mathbb{R}$ if for each $\ell \in \mathbb{Z}$, either $f(k)=f(k+\lfloor \ell r \rfloor)$ or $f(k)=f(k+\lceil \ell r \rceil)$. We say that f is ϵ-pseudoperiodic if it is pseudoperiodic for at least an ϵ fraction of the values $k=0,1,\ldots,\lfloor r \rfloor$. We require that the discretized function is ϵ-pseudoperiodic for some constant ϵ and that it is injective on the subset of inputs where it is pseudoperiodic. The periodic function encoding the regulator of Pell's equation can be constructed so that it satisfies these conditions.

The algorithm for period finding over $\mathbb{R}$ closely follows Algorithm 5. Again the basic approach is Fourier sampling over $\mathbb{Z}/N\mathbb{Z}$, with N depending on some *a priori* upper bound on the period.

Algorithm 8 (*Period finding for a pseudoperiodic function*).

Input: Black box $f:\mathbb{Z}\to S$ that is ϵ-pseudoperiodic [for some $\epsilon = \Omega(1)$] with period $r \in \mathbb{R}$.

Problem: Approximate r.

(1) Prepare the uniform superposition $|\mathbb{Z}/N\mathbb{Z}\rangle$.

(2) Query the pseudoperiodic function in an ancilla register, giving

$$\frac{1}{\sqrt{N}} \sum_{x \in \mathbb{Z}/N\mathbb{Z}} |x, f(x)\rangle. \tag{94}$$

(3) Discard the ancilla register, so that the first register is left in a uniform superposition over those x for which $f(x)$ takes some particular value. With constant probability, this is a value at which f is pseudoperiodic. Suppose that this value is $f(x_0)$, where $0 \leq x_0 \leq r$. As in step 3 of Algorithm 5, the first register is a superposition over $n \approx N/r$ points, with the rounding depending on the particular value of x_0. We write $[\ell]$ to denote an integer that could be either $\lfloor \ell \rfloor$ or $\lceil \ell \rceil$. With this notation, we obtain the state

$$\frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} |x_0 + [jr]\rangle. \tag{95}$$

(4) Perform the Fourier transform over $\mathbb{Z}/N\mathbb{Z}$, giving

$$\frac{1}{\sqrt{nN}} \sum_{k \in \mathbb{Z}/N\mathbb{Z}} \omega_N^{kx_0} \sum_{j=0}^{n-1} \omega_N^{k[jr]} |k\rangle. \tag{96}$$

We have $[jr] = jr + \delta_j$, where $-1 < \delta_j < 1$, so the sum over j above is

$$\sum_{j=0}^{n-1} \omega_N^{k[jr]} = \sum_{j=0}^{n-1} \omega_N^{kjr} \omega_N^{k\delta_j}. \tag{97}$$

When the offsets δ_j are zero, this is simply Eq. (51), which we have already shown is strongly peaked around values of k close to integer multiples of N/r. To compare with this case, we compute the deviation

$$\left|\sum_{j=0}^{n-1}\omega_N^{kjr}\omega_N^{k\delta_j}-\sum_{j=0}^{n-1}\omega_N^{kjr}\right|\leqslant\sum_{j=0}^{n-1}|\omega_N^{k\delta_j}-1|\leqslant\frac{1}{2}\sum_{j=0}^{n-1}\left|\frac{\pi k\delta_j}{N}\right|\leqslant\frac{\pi kn}{2N}. \quad (98)$$

This bound does not show that the amplitudes are close for all values of k. However, suppose we restrict our attention to those values of k less than $N/\log r$. (We obtain such a k with probability about $1/\log r$, so we can condition on such a value with only a polynomial increase in the overall running time.) Then if $k=\lfloor jN/r\rceil$ for some $j\in\mathbb{Z}$, we find [using Eq. (54)]

$$\left|\frac{1}{\sqrt{nN}}\sum_{j=0}^{n-1}\omega_N^{k[jr]}\right|=\Omega\left(\frac{1}{\sqrt{r}}\right). \quad (99)$$

(5) Measure the state of Eq. (96) in the computational basis. As in step 5 of Algorithm 5, we sample from a distribution in which some value $k=\lfloor jN/r\rceil$ (with $j\in\mathbb{Z}$) appears with reasonably large probability [now $\Omega(1/\mathrm{poly}(\log r))$ instead of $\Omega(1)$].

(6) Finally, we must obtain an approximation to r using these samples. Since r is not an integer, the procedure from step 6 of Algorithm 5 does not suffice. However, we can perform Fourier sampling sufficiently many times that we obtain two values $\lfloor jN/r\rceil$, $\lfloor j'N/r\rceil$, where j and j' are relatively prime, again with only polynomial overhead. It can be shown that if $N\geqslant 3r^2$, then j/j' is guaranteed to be one of the convergents in the continued fraction expansion of $\lfloor jN/r\rceil/\lfloor j'N/r\rceil$. Thus we can learn j and hence compute $jN/\lfloor jN/r\rceil$, which gives a good approximation to r: in particular, $|r-\lfloor jN/\lfloor jN/r\rceil\rceil|\leqslant 1$.

E. The principal ideal problem and number field cryptography

Pell's equation is closely related to another problem in algebraic number theory called the principal ideal problem. Fix a quadratic number field $\mathbb{Q}[\sqrt{d}]$ and suppose we are given an invertible ideal I, an ideal for which there exists some $J\subseteq\mathbb{Q}[\sqrt{d}]$ with $IJ=\mathbb{Z}[\sqrt{d}]$. In the principal ideal problem, we are asked to decide whether there is some $\alpha\in\mathbb{Z}[\sqrt{d}]$ such that $I=\alpha\mathbb{Z}[\sqrt{d}]$ (i.e., whether I is principal) and, if so, to find that α (or more precisely $\lfloor\log\alpha\rceil$). Notice that computing α can be viewed as an analog of the discrete logarithm problem in $\mathbb{Z}[\sqrt{d}]$. Using similar ideas as in the algorithm for solving Pell's equation and proceeding along similar lines to Algorithm 3, Hallgren (2007) also gave an efficient quantum algorithm for the principal ideal problem.

The integer factoring problem reduces to solving Pell's equation and Pell's equation reduces to the principal ideal problem (Buchmann and Williams, 1989); but no reductions in the other direction are known. Indeed, whereas factoring is conjectured to be possible with a classical computer in time $2^{O((\log d)^{1/3}(\log\log d)^{1/3})}$, the best known classical algorithms for Pell's equation and the principal ideal problem both take time $2^{O(\sqrt{\log d\log\log d})}$ assuming the generalized Riemann hypothesis or time $O(d^{1/4}\mathrm{poly}(\log d))$ with no such assumption (Buchmann, 1990; Vollmer, 2000). Motivated by the possibility that the principal ideal problem is indeed harder than factoring, Buchmann and Williams proposed a key exchange protocol based on it (Buchmann and Williams, 1989). This system is analogous to the Diffie-Hellman protocol discussed in Sec. IV.B.1, but instead of exchanging integers, Alice and Bob exchange reduced ideals. Hallgren's algorithm shows that quantum computers can efficiently break the Buchmann-Williams cryptosystem.

F. Computing the unit group of a general number field

Recall from Sec. V.B that the quantum algorithm for solving Pell's equation proceeds by computing the fundamental unit of the unit group of $\mathbb{Z}[\sqrt{d}]$. More generally, there is an efficient quantum algorithm to compute the unit group of an arbitrary number field of fixed degree (Hallgren, 2005; Schmidt and Vollmer, 2005), which we summarize.

In general, an algebraic number field (or simply number field) $\mathbb{K}=\mathbb{Q}[\theta]$ is a finite extension of the field $\mathbb{Q}$ of rational numbers. Here θ is a root of some monic irreducible polynomial over $\mathbb{Q}$ called the minimal polynomial. If the minimal polynomial has degree n, we say that $\mathbb{K}$ is a number field of degree n. For example, the quadratic number field $\mathbb{Q}[\sqrt{d}]$ has the minimal polynomial x^2-d and hence is of degree 2.

For a general number field, the units are defined as the algebraic integers of that field whose inverses are also algebraic integers. In general, the units form a group under multiplication. Just as the units of a quadratic number field are powers of some fundamental unit, it can be shown that the unit group $U(\mathbb{K})$ of any number field $\mathbb{K}$ consists of elements of the form $\zeta\epsilon_1^{n_1}\cdots\epsilon_r^{n_r}$ for $n_1,\ldots,n_r\in\mathbb{Z}$, where ζ is a root of unity and $\epsilon_1,\ldots,\epsilon_r$ are called the *fundamental units* (with r defined below). Given a number field of constant degree (say, in terms of its minimal polynomial), ζ can be computed efficiently by a classical computer. The unit group problem asks us to compute (the regulators of) the fundamental units $\epsilon_1,\ldots,\epsilon_r$.

As in the quantum algorithm for solving Pell's equation, we can reduce this computation to a period-finding problem. To see how the periodic function is defined, suppose the minimal polynomial of $\mathbb{K}$ has s real roots and t pairs of complex roots; then the number of fundamental units is $r=s+t-1$. Let $\theta_1,\ldots,\theta_s$ be the s real roots and let $\theta_{s+1},\ldots,\theta_{s+t}$ be t complex roots that, together with their complex conjugates $\theta_{s+1}^*,\ldots,\theta_{s+t}^*$, constitute all $2t$ complex roots. For each $j=1,\ldots,s+t$, we can embed $\mathbb{K}$ in $\mathbb{C}$ with the map $\sigma_j:\mathbb{K}\to\mathbb{C}$ that replaces θ by θ_j. Then we define a function $L:\mathbb{K}^\times\to\mathbb{R}^{s+t}$ as

$$L(x):=[\log|\sigma_1(x)|,\ldots,\log|\sigma_s(x)|,2\log|\sigma_{s+1}(x)|,\ldots,2\log|\sigma_{s+t}(x)|]. \quad (100)$$

By Dirichlet's theorem, $L(U(\mathbb{K}))$ is an r-dimensional lattice in $\mathbb{R}^{r+1}$ whose coordinates $(y_1,\ldots,y_{r+1})$ obey $\Sigma_j y_j=0$ (Cohen, 1993, Theorem 4.9.7). The unit group problem is essentially equivalent to finding a basis for this lattice, i.e., determining the periodicity of $L(x)$. Note that since the lattice has dimension r, we can restrict our attention to any r components of $L(x)$, thereby giving a period-finding problem over $\mathbb{R}^r$.

There are two main parts to the quantum algorithm for computing the unit group, again paralleling the algorithm for Pell's equation. First, one must show how to efficiently compute the function $L(x)$ or, more precisely, a related function that hides the same lattice, analogous to the function discussed in Sec. V.C (and again based on the concept of a reduced ideal). Second, one must generalize period finding over $\mathbb{R}$ (Sec. V.D) to period finding over $\mathbb{R}^r$. All relevant computations can be performed efficiently provided the degree of $\mathbb{K}$ is constant, giving an efficient quantum algorithm for the unit group problem in this case.

G. The principal ideal problem and the class group

We conclude our discussion of quantum algorithms for number fields by mentioning two additional problems with efficient quantum algorithms.

In Sec. V.E, we showed that the efficient quantum algorithm for Pell's equation can be adapted to decide whether a given ideal is principal and, if so, to compute (the regulator of) its generator. More generally, the principal ideal problem can be defined for any number field, and the techniques discussed in Sec. V.F can be applied to give an efficient quantum algorithm for it whenever the number field has constant degree (Hallgren, 2005).

A related problem is the task of computing the class group $\mathrm{Cl}(\mathbb{K})$ of a number field $\mathbb{K}$. The class group is defined as the set of ideals of $\mathbb{K}$ modulo the set of principal ideals of $\mathbb{K}$; it is a finite Abelian group. The class group problem asks us to decompose $\mathrm{Cl}(\mathbb{K})$ as done in Sec. IV.G. Assuming the generalized Riemann hypothesis (GRH), there is a polynomial-time algorithm to find generators of $\mathrm{Cl}(\mathbb{K})$ (Thiel, 1995). If $\mathbb{K}=\mathbb{Q}[\sqrt{-d}]$ is an imaginary quadratic number field, then its elements have unique representatives that can be computed efficiently, and $\mathrm{Cl}(\mathbb{K})$ can be decomposed using the procedure of Mosca (1999) and Cheung and Mosca (2001). More generally, it is not known how to uniquely represent the elements of $\mathrm{Cl}(\mathbb{K})$ in an efficiently computable way. However, we can take advantage of the technique introduced by Watrous (2001a) for computing over quotient groups, namely, to represent a coset by the uniform superposition of its elements. Using this idea, it can be shown that there is an efficient quantum algorithm for decomposing $\mathrm{Cl}(\mathbb{K})$, provided $\mathbb{K}$ has constant degree and assuming the GRH (Hallgren, 2005) [see Hallgren (2007) for the special case of a real quadratic number field]. In particular, we can compute $|\mathrm{Cl}(\mathbb{K})|$, the class number of the number field, $\mathbb{K}$ just as Kedlaya's algorithm does for curves (Sec. IV.H).

VI. NON-ABELIAN QUANTUM FOURIER TRANSFORM

In Sec. IV, we showed that the Abelian Fourier transform can be used to exploit the symmetry of an Abelian HSP and that this essentially gave a complete solution. In the non-Abelian version of the HSP, a non-Abelian version of the Fourier transform can similarly be used to exploit the symmetry of the problem. However, in general, this will only take us part of the way to a solution of the non-Abelian HSP.

A. The Fourier transform over a non-Abelian group

We begin by discussing the definition of the non-Abelian Fourier transform. For a more extensive review of Fourier analysis on finite groups, see Serre (1977), Diaconis (1988), and Terras (1999). Here we assume knowledge of group representation theory (see Appendix B for a summary of the requisite background).

The Fourier transform of the state $|x\rangle$ corresponding to the group element $x\in G$ is a weighted superposition over a complete set of irreducible representations $\hat{G}$, namely,

$$|\hat{x}\rangle := \frac{1}{\sqrt{|G|}}\sum_{\sigma\in\hat{G}} d_\sigma|\sigma,\sigma(x)\rangle, \tag{101}$$

where d_σ is the dimension of the representation σ, $|\sigma\rangle$ is a state that labels the irreducible representation (or irrep), and $|\sigma(x)\rangle$ is a normalized, d_σ^2-dimensional state whose amplitudes are given by the entries of the $d_\sigma\times d_\sigma$ matrix $\sigma(x)/\sqrt{d_\sigma}$,

$$|\sigma(x)\rangle := [\sigma(x)\otimes 1_{d_\sigma}]\sum_{j=1}^{d_\sigma}\frac{|j,j\rangle}{\sqrt{d_\sigma}} = \sum_{j,k=1}^{d_\sigma}\frac{\sigma(x)_{j,k}}{\sqrt{d_\sigma}}|j,k\rangle. \tag{102}$$

Here $\sigma(x)$ is a unitary matrix representing the group element $x\in G$; we have $\sigma(x)\sigma(y)=\sigma(xy)$ for all $x,y\in G$. [If σ is one dimensional, then $|\sigma(x)\rangle$ is simply a phase factor $\sigma(x)\in\mathbb{C}$ with $|\sigma(x)|=1$.] In other words, the Fourier transform over G is the unitary matrix

$$F_G := \sum_{x\in G}|\hat{x}\rangle\langle x| = \sum_{x\in G}\sum_{\sigma\in\hat{G}}\sqrt{\frac{d_\sigma}{|G|}}\sum_{j,k=1}^{d_\sigma}\sigma(x)_{j,k}|\sigma,j,k\rangle\langle x|. \tag{103}$$

Note that the Fourier transform over a non-Abelian G is not uniquely defined but rather depends on a choice of basis for each irrep of dimension greater than 1.

It is straightforward to check that F_G is indeed a unitary transformation. Using the identity

$$\langle\sigma(y)|\sigma(x)\rangle = \mathrm{Tr}[\sigma^\dagger(y)\sigma(x)]/d_\sigma = \chi_\sigma(y^{-1}x)/d_\sigma, \tag{104}$$

we have

$$\langle\hat{y}|\hat{x}\rangle = \sum_{\sigma\in\hat{G}}\frac{d_\sigma^2}{|G|}\langle\sigma(y)|\sigma(x)\rangle = \sum_{\sigma\in\hat{G}}\frac{d_\sigma}{|G|}\chi_\sigma(y^{-1}x). \tag{105}$$

Hence by Eqs. (B5) and (B6) in Appendix B, we find that $\langle\hat{y}|\hat{x}\rangle=\delta_{x,y}$.

As noted in Appendix B, F_G is precisely the transformation that simultaneously block-diagonalizes the actions of left multiplication and right multiplication or, equivalently, that decomposes both the left and right regular representations of G into their irreducible components. We check this explicitly for the left regular representation L of G. This representation satisfies $L(x)|y\rangle=|xy\rangle$ for all $x,y\in G$, so

$$\begin{aligned}\hat{L}(x) &:= F_G L(x) F_G^\dagger = \sum_{y\in G} |\widehat{xy}\rangle\langle\hat{y}| \\ &= \sum_{y\in G} \sum_{\sigma,\sigma'\in\hat{G}} \sum_{j,k=1}^{d_\sigma} \sum_{j',k'=1}^{d_{\sigma'}} \frac{\sqrt{d_\sigma d_{\sigma'}}}{|G|} \\ &\quad \times \sigma(xy)_{j,k}\sigma'(y)^*_{j',k'}|\sigma,j,k\rangle\langle\sigma',j',k'| \\ &= \sum_{y\in G} \sum_{\sigma,\sigma'\in\hat{G}} \sum_{j,k,\ell=1}^{d_\sigma} \sum_{j',k'=1}^{d_{\sigma'}} \frac{\sqrt{d_\sigma d_{\sigma'}}}{|G|} \\ &\quad \times \sigma(x)_{j,\ell}\sigma(y)_{\ell,k}\sigma'(y)^*_{j',k'}|\sigma,j,k\rangle\langle\sigma',j',k'| \\ &= \sum_{\sigma\in\hat{G}} \sum_{j,k,\ell=1}^{d_\sigma} \sigma(x)_{j,\ell}|\sigma,j,k\rangle\langle\sigma,\ell,k| \\ &= \bigoplus_{\sigma\in\hat{G}} [\sigma(x)\otimes 1_{d_\sigma}], \end{aligned} \tag{106}$$

where in the third line we have used the orthogonality relation for irreps (Theorem 5 in Appendix B).

A similar calculation can be done for the right regular representation defined by $R(x)|y\rangle=|yx^{-1}\rangle$, giving

$$\hat{R}(x) := F_G R(x) F_G^\dagger = \bigoplus_{\sigma\in\hat{G}} [1_{d_\sigma}\otimes\sigma(x)^*]. \tag{107}$$

This identity will be useful when analyzing the application of the quantum Fourier transform to the hidden subgroup problem in Sec. VII.

B. Efficient quantum circuits

In Sec. III.B, we described efficient quantum circuits for implementing the quantum Fourier transform over any finite Abelian group. Analogous circuits are known for many, but not all, non-Abelian groups. Just as the circuit for the QFT over a cyclic group parallels the usual classical fast Fourier transform (FFT), many of these circuits build on classical implementations of FFTs over non-Abelian groups (Beth, 1987; Clausen, 1989; Diaconis and Rockmore, 1990; Rockmore, 1990; Maslen and Rockmore, 1995). Here we summarize the groups for which efficient QFTs are known.

Høyer (1997) gave efficient circuits for the quantum Fourier transform over metacyclic groups (i.e., semidirect products of cyclic groups), including the dihedral group, and over the Pauli group on n qubits. An alternative construction for certain metacyclic two-groups has been given by Püschel *et al.* (1999). Beals (1997) gave an efficient implementation of the QFT over the symmetric group. Finally, Moore *et al.* (2006) gave a general construction of QFTs, systematically quantizing classical FFTs. For example, this approach yields polynomial-time quantum circuits for Clifford groups, the symmetric group, the wreath product of a polynomial-sized group, and metabelian groups.

There are a few important groups for which efficient quantum Fourier transforms are not known. These include the classical groups, such as the group $\mathsf{GL}_n(\mathbb{F}_q)$ of $n\times n$ invertible matrices over a finite field with q elements. However, it is possible to implement these transforms in subexponential time (Moore *et al.*, 2006).

VII. NON-ABELIAN HIDDEN SUBGROUP PROBLEM

We now turn to the general non-Abelian version of the hidden subgroup problem. We begin by stating the problem and describing some of its potential applications. We describe the standard way of approaching the problem on a quantum computer and explain how the non-Abelian Fourier transform can be used to simplify the resulting hidden subgroup states. This leads to the notions of weak Fourier sampling and strong Fourier sampling; we describe some of their applications and limitations. Then we discuss how multiregister measurements on hidden subgroup states can potentially avoid some of those limitations. Finally, we describe two specific algorithmic techniques for hidden subgroup problems: the Kuperberg sieve and the pretty good measurement. Note that some of the results presented in Sec. VIII, on the hidden shift problem, also give algorithms for the non-Abelian HSP.

A. The problem and its applications

The non-Abelian hidden subgroup problem naturally generalizes the Abelian HSP considered in Sec. IV. In the hidden subgroup problem for a group G, we are given a black-box function $f:G\to S$, where S is a finite set. We say that f hides a subgroup $H\leqslant G$ provided

$$f(x)=f(y) \text{ if and only if } x^{-1}y\in H \tag{108}$$

(where we use multiplicative notation for non-Abelian groups). In other words, f is constant on left cosets $H, g_1H, g_2H,\ldots$ of H in G and distinct on different left cosets. We say that an algorithm for the HSP in G is efficient if it runs in time $\mathrm{poly}(\log|G|)$.

The choice of left cosets is an arbitrary one; we could just as well define the HSP in terms of right cosets $H, Hg_1', Hg_2',\ldots$ by promising that $f(x)=f(y)$ if and only if $xy^{-1}\in H$. But here we use the definition in terms of left cosets.

The non-Abelian HSP is of interest not only because it generalizes the Abelian case in a natural way but because a solution of certain non-Abelian HSPs would have particularly useful applications. The most well known (and also the most straightforward) applications are to the graph automorphism problem and the graph

isomorphism problem (Boneh and Lipton, 1995; Beals, 1997; Høyer, 1997; Ettinger and Høyer, 1999).

In the graph automorphism problem, we are given a graph Γ on n vertices, and our goal is to determine its automorphism group. We say that $\pi \in S_n$ is an automorphism of Γ if $\pi(\Gamma)=\Gamma$. The automorphisms of Γ form a group $\mathrm{Aut}\Gamma \leqslant S_n$; if $\mathrm{Aut}\Gamma$ is trivial then we say Γ is *rigid*. We may cast the graph automorphism problem as an HSP over S_n by considering the function $f(\pi) := \pi(\Gamma)$, which hides $\mathrm{Aut}\Gamma$.

In the graph isomorphism problem, we are given two connected graphs Γ, Γ' each on n vertices, and our goal is to determine whether there is any permutation $\pi \in S_n$ such that $\pi(\Gamma)=\Gamma'$, in which case we say that Γ and Γ' are *isomorphic*. We can cast graph isomorphism as an HSP in the wreath product $S_n \wr S_2 \leqslant S_{2n}$. (The wreath product group $G \wr T$, where $T \leqslant S_m$, is the semidirect product $G^m \rtimes T$, where T acts to permute the elements of G^m.) Writing the elements of $S_n \wr S_2$ in the form (σ, τ, b) where $\sigma, \tau \in S_n$ represent permutations of Γ, Γ', respectively, and $b \in \{0,1\}$ denotes whether to swap the two graphs, we define

$$f(\sigma,\tau,b) := \begin{cases} (\sigma(\Gamma), \tau(\Gamma')), & b=0, \\ (\sigma(\Gamma'), \tau(\Gamma)), & b=1. \end{cases} \tag{109}$$

The function f hides the automorphism group of the disjoint union of Γ and Γ'. This group contains an element that swaps the two graphs and hence is at least twice as large as $|\mathrm{Aut}\Gamma| \cdot |\mathrm{Aut}\Gamma'|$ if and only if the graphs are isomorphic. In particular, if Γ and Γ' are rigid [which seems to be a hard case for the HSP approach to graph isomorphism and in fact is equivalent to the problem of deciding rigidity (Hoffmann, 1982)], the hidden subgroup is trivial when Γ, Γ' are nonisomorphic and has order 2, with its nontrivial element the involution $(\pi, \pi^{-1}, 1)$, when $\Gamma = \pi(\Gamma')$.

The graph automorphism and graph isomorphism problems are closely related. The decision version of graph isomorphism is polynomial-time equivalent to the problems of finding an isomorphism between two graphs provided one exists, counting the number of such isomorphisms, finding the automorphism group of a single graph, and computing the size of this automorphsim group (Hoffmann, 1982). Deciding whether a graph is rigid (i.e., whether the automorphsim group is trivial) can be reduced to general graph isomorphism, but the other direction is unknown, so deciding rigidity could be an easier problem (Köbler *et al.*, 1993).

We point out the possibility that graph isomorphism is not a hard problem even for classical computers. There are polynomial-time classical algorithms for many special cases of graph isomorphism, such as when the maximum degree is bounded (Luks, 1982), the genus is bounded (Filotti and Mayer, 1980; Miller, 1980), or the eigenvalue multiplicity is bounded (Babai *et al.*, 1982). Furthermore, there are classical algorithms that run in time $2^{O(\sqrt{n \log n})}$ for general graphs (Babai *et al.*, 1983) and in time $2^{O(\sqrt[3]{n})}$ for strongly regular graphs (Spielman, 1996), which are suspected to be some of the hardest graphs for the problem. Even if there is a polynomial-time quantum algorithm for graph isomorphism, it is plausible that the HSP in the symmetric group might be substantially harder since the graph structure is lost in the reduction to the HSP. Indeed, solving the HSP in the symmetric group would equally well solve other isomorphism problems, such as the problem of code equivalence (Ettinger and Høyer, 1999), which is at least as hard as graph isomorphism and possibly harder (Petrank and Roth, 1997).

The second major potential application of the hidden subgroup problem is to lattice problems. An n-dimensional lattice is the set of all integer linear combinations of n linearly independent vectors in $\mathbb{R}^n$ (a basis for the lattice). In the shortest vector problem, we are asked to find a shortest nonzero vector in the lattice [see, e.g., Micciancio and Goldwasser (2002)]. In particular, in the $g(n)$-unique shortest vector problem, we are promised that the shortest nonzero vector is unique (up to its sign) and is shorter than any other nonparallel vector by a factor $g(n)$. This problem can be solved in polynomial time on a classical computer if $g(n) = (1+\epsilon)^{\Omega(n)}$ (Lenstra *et al.*, 1982) and even if $g(n) = 2^{\Omega(n \log\log n/\log n)}$ (Schnorr, 1987; Ajtai *et al.*, 2001). The problem is NP-hard if $g(n)=O(1)$ (van Emde Boas, 1981; Ajtai, 1998; Micciancio, 2001); in fact, even stronger hardness results are known (Khot, 2005). For $g(n) = \mathrm{poly}(n)$, the problem is suspected to be hard, at least for a classical computer. In particular, the presumed hardness of the $O(n^8)$-unique shortest vector problem is the basis for a cryptosystem proposed by Ajtai (1996), Ajtai and Dwork (1997), and Micciancio and Goldwasser (2002), and a subsequent improvement by Regev (2004a) requires quantum hardness of the $O(n^{1.5})$-unique shortest vector problem.

Regev showed that an efficient quantum algorithm for the dihedral hidden subgroup problem based on the standard method (described below) could be used to solve the poly(n)-unique shortest vector problem (Regev, 2004b). Such an algorithm would be significant since it would break these lattice cryptosystems, which are some of the few proposed cryptosystems that are not compromised by Shor's algorithm.

So far, only the symmetric and dihedral hidden subgroup problems are known to have applications to natural problems. Nevertheless, there has been considerable interest in understanding the complexity of the HSP for general groups. There are at least three reasons for this. First, the problem is simply of fundamental interest: it appears to be a natural setting for exploring the extent of the advantage of quantum computers over classical ones. Second, techniques developed for other HSPs may eventually find application to the symmetric or dihedral groups. Finally, exploring the limitations of quantum computers for HSPs may suggest cryptosystems that could be robust even to quantum attacks (Okamoto *et al.*, 2000; Regev, 2004b; Kawachi *et al.*, 2005; Moore, Russell, and Vazirani, 2007; Hayashi *et al.*, 2008).

B. The standard method

Nearly all known algorithms for the non-Abelian hidden subgroup problem use the black box for f in essentially the same way as in the Abelian HSP (Sec. IV.C). This approach has therefore come to be known as the standard method.

In the standard method, we begin by preparing a uniform superposition over group elements,

$$|G\rangle := \frac{1}{\sqrt{|G|}} \sum_{x \in G} |x\rangle. \tag{110}$$

We then compute the value $f(x)$ in an ancilla register, giving

$$\frac{1}{\sqrt{|G|}} \sum_{x \in G} |x, f(x)\rangle. \tag{111}$$

Finally, we discard the second register. If we were to measure the second register, obtaining the outcome $y \in S$, then the state would be projected onto the uniform superposition of those $x \in G$ such that $f(x)=y$, which is simply some left coset of H. Since every coset contains the same number of elements, each left coset occurs with equal probability. Thus discarding the second register yields the coset state

$$|xH\rangle := \frac{1}{\sqrt{|H|}} \sum_{h \in H} |xh\rangle \quad \text{with } x \in G \text{ uniformly random and unknown.} \tag{112}$$

Depending on context, it may be more convenient to view the outcome either as a random pure state or, equivalently, as the mixed quantum state

$$\rho_H := \frac{1}{|G|} \sum_{x \in G} |xH\rangle\langle xH|, \tag{113}$$

which we refer to as a hidden subgroup state. In the standard approach to the hidden subgroup problem, we attempt to determine H using samples of this hidden subgroup state.

Historically, work on the hidden subgroup problem has focused almost exclusively on the standard method. However, while this method seems quite natural, there is no general proof that it is necessarily the best way to approach the HSP. Koiran *et al.* (2007) showed that the quantum query complexity of Simon's problem is linear, so that Simon's algorithm (using the standard method) is within a constant factor of optimal. This immediately implies an $\Omega(n)$ lower bound for the HSP in any group that contains the subgroup $(\mathbb{Z}/2\mathbb{Z})^n$. It would be interesting to prove similar results for more general groups or to find other ways of evaluating the effectiveness of the standard method as compared with more general strategies.

C. Weak Fourier sampling

The symmetry of a coset state [Eq. (112)] and, equivalently, a hidden subgroup state [Eq. (113)] can be exploited using the quantum Fourier transform. In particular, we have

$$|xH\rangle = \frac{1}{\sqrt{|H|}} \sum_{h \in H} R(h)|x\rangle, \tag{114}$$

where R is the right regular representation of G. Thus the hidden subgroup state can be written as

$$\begin{aligned} \rho_H &= \frac{1}{|G| \cdot |H|} \sum_{x \in G} \sum_{h,h' \in H} R(h)|x\rangle\langle x|R(h')^\dagger \\ &= \frac{1}{|G| \cdot |H|} \sum_{h,h' \in H} R(hh'^{-1}) = \frac{1}{|G|} \sum_{h \in H} R(h). \end{aligned} \tag{115}$$

Since the right regular representation is block-diagonal in the Fourier basis, the same is true of ρ_H. In particular, using Eq. (126), we have

$$\hat{\rho}_H := F_G \rho_H F_G^\dagger = \frac{1}{|G|} \bigoplus_{\sigma \in \hat{G}} [I_{d_\sigma} \otimes \sigma(H)^*], \tag{116}$$

where

$$\sigma(H) := \sum_{h \in H} \sigma(h). \tag{117}$$

Since $\hat{\rho}_H$ is block-diagonal, with blocks labeled by irreducible representations, we may now measure the irrep label without loss of information. This procedure is referred to as weak Fourier sampling. The probability of observing representation $\sigma \in \hat{G}$ under weak Fourier sampling is

$$\Pr(\sigma) = \frac{1}{|G|} \mathrm{Tr}[I_{d_\sigma} \otimes \sigma(H)^*] = \frac{d_\sigma}{|G|} \sum_{h \in H} \chi_\sigma(h)^*, \tag{118}$$

which is precisely $d_\sigma |H|/|G|$ times the number of times the trivial representation appears in $\mathrm{Res}^G_H \sigma$, the restriction of σ to H (Hallgren *et al.*, 2003, Theorem 1.2). We may now ask whether polynomially many samples from this distribution are sufficient to determine H, and if so whether H can be reconstructed from this information efficiently.

If G is Abelian, then all of its representations are one dimensional, so weak Fourier sampling reveals all of the available information about ρ_H. This information can indeed be used to efficiently determine H, as discussed in Sec. IV.C.

Weak Fourier sampling succeeds for a similar reason whenever H is a normal subgroup of G (denoted $H \trianglelefteq G$), i.e., whenever $gHg^{-1}=H$ for all $g \in G$ (Hallgren *et al.*, 2003). In this case, the hidden subgroup state within the irrep $\sigma \in \hat{G}$ is proportional to

$$\sigma(H)^* = \frac{1}{|G|}\sum_{g\in G, h\in H} \sigma(ghg^{-1})^*, \tag{119}$$

and this commutes with $\sigma(x)^*$ for all $x \in G$, so by Schur's Lemma (Theorem 4) it is a multiple of the identity. Thus $\hat{\rho}_H$ is proportional to the identity within each block, and again weak Fourier sampling reveals all available information about H. Indeed, the distribution under weak Fourier sampling is particularly simple: we have

$$\Pr(\sigma) = \begin{cases} d_\sigma^2 |H|/|G| & \text{if } H \subseteq \ker \sigma \\ 0 & \text{otherwise} \end{cases} \tag{120}$$

(a straightforward generalization of the distribution seen in step 5 of Algorithm 4), where $\ker \sigma := \{g \in G : \sigma(g) = 1\}$ denotes the *kernel* of the representation σ. To see this, note that if $H \subseteq \ker \sigma$, then there is some $h' \in H$ with $\sigma(h') \neq 1$; but then $\sigma(h')\sigma(H) = \sum_{h\in H}\sigma(h'h) = \sigma(H)$, and since $\sigma(h')$ is unitary and $\sigma(H)$ is a scalar multiple of the identity, this can only be satisfied if in fact $\sigma(H) = 0$. On the other hand, if $H \subseteq \ker \sigma$, then $\chi_\sigma(h) = d_\sigma$ for all $h \in H$, and the result is immediate.

To find H, we can simply proceed as in the Abelian case.

Algorithm 9 (*Finding a normal hidden subgroup*).

Input: Black-box function hiding $H \trianglelefteq G$.

Problem: Determine H.

(1) Let $K_0 := G$. For $t = 1, \ldots, T$, where $T = O(\log|G|)$,

(a) perform weak Fourier sampling, obtaining an irrep $\sigma_t \in \hat{G}$;

(b) let $K_t := K_{t-1} \cap \ker \sigma_t$.

(2) Output K_T.

To see that this works, suppose that at the tth step the intersection of the kernels is $K_{t-1} \trianglelefteq G$ with $K_{t-1} \neq H$ (so that, in particular, $|K_{t-1}| \geq 2|H|$); then the probability of obtaining an irrep σ for which $K_{t-1} \subseteq \ker \sigma$ is (cf. step 6 of Algorithm 4)

$$\frac{|H|}{|G|}\sum_{\sigma: K_{t-1}\subseteq \ker \sigma} d_\sigma^2 = \frac{|H|}{|K_{t-1}|} \leq \frac{1}{2}, \tag{121}$$

where we have used the fact that the distribution in Eq. (120) remains normalized if H is replaced by any normal subgroup of G. Each repetition of weak Fourier sampling has a probability of at least $1/2$ of cutting the intersection of the kernels at least in half, so we converge to H in $O(\log|G|)$ steps. In fact, applying the same approach when H is not necessarily normal in G gives an algorithm to find the *normal core* of H, the largest subgroup of H that is normal in G (Hallgren *et al.*, 2003).

This algorithm can be applied to find hidden subgroups in groups that are "close to Abelian" a certain sense. In particular, Grigni *et al.* (2004) showed that if $\kappa(G)$, the intersection of the normalizers of all subgroups of G, is sufficiently large—specifically, if $|G|/|\kappa(G)| = 2^{O(\log^{1/2} n)}$, such as when $G = \mathbb{Z}/3\mathbb{Z} \rtimes \mathbb{Z}/2^n\mathbb{Z}$—then the HSP in G can be solved in polynomial time. The idea is simply to apply the algorithm for normal subgroups to all subgroups containing $\kappa(G)$; the union of all subgroups obtained in this way gives the hidden subgroup with high probability. This result was subsequently improved to give a polynomial-time quantum algorithm whenever $|G|/|\kappa(G)| = \text{poly}(\log|G|)$ (Gavinsky, 2004).

D. Strong Fourier sampling

Despite the examples given in the previous section, weak Fourier sampling does not provide sufficient information to recover the hidden subgroup in the majority of non-Abelian hidden subgroup problems. For example, weak Fourier sampling fails to solve the HSP in the symmetric group (Hallgren *et al.*, 2003; Grigni *et al.*, 2004) and the dihedral group.

To obtain more information about the hidden subgroup, we can perform a measurement on the d_σ^2-dimensional state that results when weak Fourier sampling returns the outcome σ. Such an approach is referred to as *strong Fourier sampling*. From Eq. (116), this d_σ^2-dimensional state is the tensor product of a d_σ-dimensional maximally mixed state for the row register (as a consequence of the fact that the left and right regular representations commute) with some d_σ-dimensional state $\hat{\rho}_{H,\sigma}$ for the column register. Since the row register does not depend on H, we may discard this register without loss of information. In other words, strong Fourier sampling is effectively faced with the state

$$\hat{\rho}_{H,\sigma} = \frac{\sigma(H)^*}{\sum_{h\in H}\chi_\sigma(h)^*}. \tag{122}$$

This state is proportional to a projector whose rank is simply the number of times the trivial representation appears in $\text{Res}_H^G \sigma^*$. This follows because

$$\sigma(H)^2 = \sum_{h,h'\in H}\sigma(hh') = |H|\sigma(H), \tag{123}$$

which gives

$$\hat{\rho}_{H,\sigma}^2 = \frac{|H|}{\sum_{h\in H}\chi_\sigma(h)^*}\hat{\rho}_{H,\sigma}, \tag{124}$$

so that $\hat{\rho}_{H,\sigma}$ is proportional to a projector with $\text{rank}(\hat{\rho}_{H,\sigma}) = \sum_{h\in H}\chi_\sigma(h)^*/|H|$.

It is not immediately clear how to choose a good basis for strong Fourier sampling, so a natural first approach is to consider the effect of measuring in a random basis (i.e., a basis chosen uniformly with respect to the Haar measure over $\mathbb{C}^{d_\sigma}$). There are a few cases in which such *random strong Fourier sampling* is fruitful. For example, Radhakrishnan *et al.* (2009) showed that measuring in a random basis provides sufficient information to solve the HSP in the Heisenberg group. Subsequently, Sen (2006) generalized this result to show that random strong Fourier sampling is information-theoretically sufficient whenever $\text{rank}(\hat{\rho}_{H,\sigma}) = \text{poly}(\log|G|)$ for all $\sigma \in \hat{G}$ (for

example, when G is the dihedral group), as a consequence of a more general result on the distinguishability of quantum states using random measurements.

In some cases random strong Fourier sampling is unhelpful. For example, Grigni *et al.* (2004) showed that if H is sufficiently small and G is sufficiently non-Abelian (in a certain precise sense), then random strong Fourier sampling is not very informative. In particular, they showed this for the problem of finding hidden involutions in the symmetric group. Another example was provided by Moore, Rockmore, *et al.* (2007), who showed that random strong Fourier sampling fails in the metacyclic groups $\mathbb{Z}/p\mathbb{Z}\rtimes\mathbb{Z}/q\mathbb{Z}$ [subgroups of the affine group $\mathbb{Z}/p\mathbb{Z}\rtimes(\mathbb{Z}/p\mathbb{Z})^\times$] when $q<p^{1-\epsilon}$ for some $\epsilon>0$.

Even when measuring in a random basis is information-theoretically sufficient, it does not give an efficient quantum algorithm; we must consider both the implementation of the measurement and the interpretation of its outcomes. We cannot measure in a random basis, but we can instead try to find explicit bases in which strong Fourier sampling can be performed efficiently, and for which the results solve the HSP. The first such algorithm was provided by Moore, Rockmore, *et al.* (2007) for the metacyclic groups $\mathbb{Z}/p\mathbb{Z}\rtimes\mathbb{Z}/q\mathbb{Z}$ with $q=p/\mathrm{poly}(\log p)$. Note that for these values of p,q, unlike the case $q<p^{1-\epsilon}$ mentioned above, measurement in a random basis is information-theoretically sufficient. Indeed, we do not know of any example of an HSP for which strong Fourier sampling gives an efficient algorithm yet random strong Fourier sampling fails information theoretically; it would be interesting to find any such example (or to prove that none exists).

Of course, simply finding an informative basis is not sufficient; it is also important that the measurement results can be efficiently postprocessed. This issue arises not only in the context of measurement in a pseudorandom basis but also in the context of certain explicit bases. For example, Ettinger and Høyer (2000) gave a basis for the dihedral HSP in which a measurement gives sufficient classical information to infer the hidden subgroup, but no efficient means of postprocessing this information is known (see Sec. VIII.A).

For some groups, it turns out that strong Fourier sampling simply fails. Moore *et al.* (2008) showed that, regardless of what basis is chosen, strong Fourier sampling provides insufficient information to solve the HSP in the symmetric group. Specifically, they showed that for any measurement basis (indeed, for any generalized measurement on the hidden subgroup states), the distributions of outcomes in the cases where the hidden subgroup is trivial and where the hidden subgroup is a random involution are exponentially close.

E. Multiregister measurements and query complexity

Even if we restrict our attention to the standard method, the failure of strong Fourier sampling does not necessarily mean that the HSP cannot be solved. In general, we need not restrict ourselves to measurements acting on a single hidden subgroup state ρ_H at a time; rather, it may be advantageous to measure joint observables on $\rho_H^{\otimes k}$ for $k>1$. Such an approach could conceivably be efficient provided $k=\mathrm{poly}(\log|G|)$.

By considering joint measurements of many hidden subgroup states at a time, Ettinger *et al.* (1999, 2004) showed that the *query complexity* of the HSP is polynomial. In other words, $\mathrm{poly}(\log|G|)$ queries of the black-box function f suffice to determine H. Unfortunately, this does not necessarily mean that the (quantum) *computational complexity* of the HSP is polynomial since it is not clear in general how to perform the quantum postprocessing of $\rho_H^{\otimes\mathrm{poly}(\log|G|)}$ efficiently. Nevertheless, this is an important observation since it already shows a difference between quantum and classical computation: recall that the classical query complexity of even the Abelian HSP is typically exponential. Furthermore, it offers some clues as to how we might design efficient algorithms.

To show that the query complexity of the HSP is polynomial, it is sufficient to show that the (single-copy) hidden subgroup states are pairwise statistically distinguishable, as measured by the quantum fidelity,

$$F(\rho,\rho') := \mathrm{Tr}|\sqrt{\rho}\sqrt{\rho'}|. \tag{125}$$

This follows from a result of Barnum and Knill (2002), who showed the following.

Theorem 3. Suppose ρ is drawn from an ensemble $\{\rho_1,\ldots,\rho_N\}$, where each ρ_i occurs with some fixed prior probability p_i. Then there exists a quantum measurement (the pretty good measurement[10]) that identifies ρ with probability at least

$$1-N\sqrt{\max_{i\neq j}F(\rho_i,\rho_j)}. \tag{126}$$

In fact, by the minimax theorem, this holds even without assuming a prior distribution for the ensemble (Harrow and Winter, 2006).

Given only one copy of the hidden subgroup state, Eq. (126) will typically give a trivial bound. However, by taking multiple copies of the hidden subgroup states, we can ensure that the overall states are nearly orthogonal and hence distinguishable. In particular, since $F(\rho^{\otimes k},\rho'^{\otimes k})=F(\rho,\rho')^k$, arbitrarily small error probability $\epsilon>0$ can be achieved using

$$k\geqslant\left\lceil\frac{2(\log N-\log\varepsilon)}{\log(1/\max_{i\neq j}F(\rho_i,\rho_j))}\right\rceil \tag{127}$$

copies of ρ.

Provided that G does not have too many subgroups and that the fidelity between two distinct hidden subgroup states is not too close to 1, this shows that polynomially many copies of ρ_H suffice to solve the HSP. The

[10]To distinguish the states ρ_i with prior probabilities p_i, the pretty good measurement (PGM) uses the measurement operators $E_i:=p_i\varrho^{-1/2}\rho_i\varrho^{-1/2}$, where $\varrho:=\Sigma_i p_i\rho_i$ [see, e.g., Hausladen and Wootters (1994)].

total number of subgroups of G is $2^{O(\log^2|G|)}$, which can be seen as follows. Any group K can be specified in terms of at most $\log_2|K|$ generators since every additional (nonredundant) generator increases the size of the group by at least a factor of 2. Since every subgroup of G can be specified by a subset of at most $\log_2|G|$ elements of G, the number of subgroups of G is upper bounded by $|G|^{\log_2|G|}=2^{(\log_2|G|)^2}$. Thus $k=\text{poly}(\log|G|)$ copies of ρ_H suffice to solve the HSP provided the maximum fidelity is bounded away from 1 by at least $1/\text{poly}(\log|G|)$.

To upper bound the fidelity between two states ρ and ρ', let Π_ρ denote the projector onto the support of ρ. By considering the projective measurement with elements Π_ρ and $1-\Pi_\rho$ and noting that the classical fidelity of the resulting distribution is an upper bound on the quantum fidelity, we have

$$F(\rho,\rho') \leqslant \sqrt{\text{Tr}\,\Pi_\rho\rho'}. \tag{128}$$

Now consider the fidelity between ρ_H and $\rho_{H'}$ for two distinct subgroups $H,H'\leqslant G$. Let $|H|\geqslant|H'|$ without loss of generality. We can write Eq. (113) as

$$\rho_H = \frac{|H|}{|G|}\sum_{x\in T_H}|xH\rangle\langle xH|, \tag{129}$$

where T_H is a left transversal of H (i.e., a complete set of unique representatives for the left cosets of H in G). Since Eq. (129) is a spectral decomposition of ρ_H, we have

$$\Pi_{\rho_H} = \sum_{x\in T_H}|xH\rangle\langle xH| = \frac{1}{|H|}\sum_{x\in G}|xH\rangle\langle xH|. \tag{130}$$

Then we have

$$\begin{aligned} F(\rho_H,\rho_{H'})^2 &\leqslant \text{Tr}\,\Pi_{\rho_H}\rho_{H'} \\ &= \frac{1}{|H|\cdot|G|}\sum_{x,x'\in G}|\langle xH|x'H'\rangle|^2 \\ &= \frac{1}{|H|\cdot|G|}\sum_{x,x'\in G}\frac{|xH\cap x'H'|^2}{|H|\cdot|H'|} \\ &= \frac{|H\cap H'|}{|H|} \leqslant \tfrac{1}{2}, \end{aligned} \tag{131}$$

where we have used the fact that

$$|xH\cap x'H'| = \begin{cases} |H\cap H'| & \text{if } x^{-1}x'\in HH' \\ 0 & \text{otherwise} \end{cases} \tag{132}$$

to evaluate

$$\begin{aligned} \sum_{x,x'\in G}|xH\cap xH'|^2 &= |G|\cdot|H\cap H'|^2\cdot|HH'| \\ &= |G|\cdot|H|\cdot|H'|\cdot|H\cap H'|. \end{aligned} \tag{133}$$

This shows that $F(\rho_H,\rho_{H'})\leqslant 1/\sqrt{2}$, thereby establishing that the query complexity of the HSP is $\text{poly}(\log|G|)$.

It is possible to obtain tighter bounds on the number of hidden subgroup states needed to solve the HSP. For example, Bacon *et al.* (2006) showed that $(1+o(1))\log_2 N$ hidden subgroup states are necessary and sufficient to find a hidden reflection in the dihedral group of order $2N$. In a similar vein, Hayashi *et al.* (2008) gave asymptotically tight bounds on the number of hidden subgroup states needed to solve the HSP in general groups, taking into account both the number of candidate subgroups and their sizes.

The measurements described here are highly multiregister: they observe correlated properties of all of $\text{poly}(\log|G|)$ hidden subgroup states at once. Thus they are quite far from strong Fourier sampling, in which measurements are made on only one hidden subgroup state at a time. It is natural to ask whether some less entangled measurement might also be sufficient for general groups, perhaps measuring a smaller number of hidden subgroup states at a time and adaptively using those measurement results to decide what measurements to make on successive hidden subgroup states. However, Hallgren *et al.* (2006) showed that this is not always the case: in the symmetric group (as well as a few other groups such as the general linear group), entangled measurements on $\Omega(\log|G|)$ registers at a time are required to solve the HSP.

F. The Kuperberg sieve

In this section, we describe an approach developed by Kuperberg (2005) that gives a subexponential (though not polynomial) time algorithm for the dihedral hidden subgroup problem—specifically, it runs in time $2^{O(\sqrt{\log|G|})}$.

The dihedral group of order $2N$, denoted D_N, is the group of symmetries of a regular N-gon. It has the presentation

$$D_N = \langle r,s|r^2=s^N=1, rsr=s^{-1}\rangle. \tag{134}$$

Here r can be viewed as a reflection about some fixed axis and s can be viewed as a rotation by an angle $2\pi/N$.

Using the defining relations, we can write any group element in the form s^xr^a, where $x\in\mathbb{Z}/N\mathbb{Z}$ and $a\in\mathbb{Z}/2\mathbb{Z}$. Thus we can equivalently think of the group as consisting of elements $(x,a)\in\mathbb{Z}/N\mathbb{Z}\times\mathbb{Z}/2\mathbb{Z}$. Since

$$(s^xr^a)(s^yr^b) = s^xr^as^yr^ar^{a+b} = s^xs^{(-1)^ay}r^{a+b} = s^{x+(-1)^ay}r^{a+b}, \tag{135}$$

the group operation for such elements can be expressed as

$$(x,a)\cdot(y,b) = (x+(-1)^ay, a+b). \tag{136}$$

[In particular, this shows that the dihedral group is the semidirect product $\mathbb{Z}/N\mathbb{Z} \rtimes_\varphi \mathbb{Z}/2\mathbb{Z}$, where $\varphi: \mathbb{Z}/2\mathbb{Z} \to \mathrm{Aut}(\mathbb{Z}/N\mathbb{Z})$ is defined by $\varphi(a)(y) = (-1)^a y$.] It is also easy to see that the group inverse is

$$(x,a)^{-1} = (-(-1)^a x, a). \tag{137}$$

The subgroups of D_N are either cyclic or dihedral. The subgroups that are cyclic are of the form $\langle (x,0) \rangle$, where $x \in \mathbb{Z}/N\mathbb{Z}$ is some divisor of N (including $x=N$). The subgroups that are dihedral are of the form $\langle (x,0), (y,1) \rangle$, where $x \in \mathbb{Z}/N\mathbb{Z}$ is some divisor of N and $y \in \mathbb{Z}/x\mathbb{Z}$; in particular, there are subgroups of the form $\langle (y,1) \rangle$, where $y \in \mathbb{Z}/N\mathbb{Z}$. A result of Ettinger and Høyer (2000) reduces the general dihedral HSP, in which the hidden subgroup could be any of these possibilities, to the dihedral HSP with the promise that the hidden subgroup is of the form $\langle (y,1) \rangle = \{(0,0), (y,1)\}$, i.e., a subgroup of order 2 generated by the reflection $(y,1)$.[11] Thus, from now on we assume that the hidden subgroup is of the form $\langle (y,1) \rangle$ for some $y \in \mathbb{Z}/N\mathbb{Z}$ without loss of generality.

When the hidden subgroup is $H = \langle (y,1) \rangle$, one particular left transversal of H in G consists of the left coset representatives $(z,0)$ for all $z \in \mathbb{Z}/N\mathbb{Z}$. The coset state [Eq. (112)] corresponding to the coset $(z,0)H$ is

$$|(z,0)H\rangle = \frac{1}{\sqrt{2}}(|z,0\rangle + |y+z,1\rangle). \tag{138}$$

We showed in Sec. VII.C that to distinguish coset states, in general, one should start with weak Fourier sampling: apply a Fourier transform over G and then measure the irrep label. Equivalently, we can simply Fourier transform the first register over $\mathbb{Z}/N\mathbb{Z}$, leaving the second register alone. When the resulting measurement outcome k is not 0 or $N/2$, this procedure is effectively the same as performing weak Fourier sampling, obtaining a two-dimensional irrep labeled by either k (for $k \in \{1, \ldots, \lceil N/2 \rceil - 1\}$) or $-k$ (for $k \in \{\lfloor N/2 \rfloor + 1, \ldots, N-1\}$), with the uniformly random sign of k corresponding to the maximally mixed row index and the remaining qubit state corresponding to the column index. For $k = 0$ or $N/2$, the representation is reducible, corresponding to a pair of one-dimensional representations.

Fourier transforming the first register over $\mathbb{Z}/N\mathbb{Z}$, we obtain

$$\begin{aligned}(F_{\mathbb{Z}/N\mathbb{Z}} \otimes I_2)|(z,0)H\rangle &= \frac{1}{\sqrt{2N}} \sum_{k \in \mathbb{Z}/N\mathbb{Z}} (\omega_N^{kz}|k,0\rangle \\ &\quad + \omega_N^{k(y+z)}|k,1\rangle) \\ &= \frac{1}{\sqrt{N}} \sum_{k \in \mathbb{Z}/N\mathbb{Z}} \omega_N^{kz}|k\rangle \\ &\quad \otimes \frac{1}{\sqrt{2}}(|0\rangle + \omega_N^{ky}|1\rangle).\end{aligned} \tag{139}$$

If we then measure the first register, we obtain one of the N values of k uniformly at random, and we are left with the postmeasurement state

$$|\psi_k\rangle := \frac{1}{\sqrt{2}}(|0\rangle + \omega_N^{yk}|1\rangle) \tag{140}$$

(dropping an irrelevant global phase that depends on z). Thus we are left with the problem of determining y given the ability to produce single-qubit states $|\psi_k\rangle$ of this form (where k is known). Since this procedure is equivalent to dihedral weak Fourier sampling, there is no loss of information in processing the state to produce Eq. (140).

It would be useful if we could prepare states $|\psi_k\rangle$ with particular values of k. For example, given the state $|\psi_{N/2}\rangle = (|0\rangle + (-1)^y|1\rangle)/\sqrt{2}$, we can learn the parity of y (i.e., its least significant bit) by measuring in the basis of states $|\pm\rangle := (|0\rangle \pm |1\rangle)/\sqrt{2}$. The main idea of Kuperberg's algorithm is to combine states of the form Eq. (140) to produce new states of the same form but with more desirable values of k.

To combine states, we can use the following procedure. Given two states $|\psi_p\rangle$ and $|\psi_q\rangle$, perform a controlled-not gate from the former to the latter, giving

$$\begin{aligned}|\psi_p, \psi_q\rangle &= \tfrac{1}{2}(|0,0\rangle + \omega_N^{yp}|1,0\rangle + \omega_N^{yq}|0,1\rangle + \omega_N^{y(p+q)}|1,1\rangle) \\ &\mapsto \tfrac{1}{2}(|0,0\rangle + \omega_N^{yp}|1,1\rangle + \omega_N^{yq}|0,1\rangle + \omega_N^{y(p+q)}|1,0\rangle) \\ &= \frac{1}{\sqrt{2}}(|\psi_{p+q},0\rangle + \omega_N^{yq}|\psi_{p-q},1\rangle).\end{aligned} \tag{141}$$

Then a measurement on the second qubit leaves the first qubit in the state $|\psi_{p\pm q}\rangle$ (up to an irrelevant global phase), with the + sign occurring when the outcome is 0 and the − sign occurring when the outcome is 1, each outcome occurring with probability 1/2.

Note that this combination procedure can be viewed as implementing the *Clebsch-Gordan decomposition*, the decomposition of a tensor product of representations into its irreducible constituents. The state indices p and q can be interpreted as labels of irreps of D_N, and the extraction of $|\psi_{p\pm q}\rangle$ can be seen as transforming their tensor product (a reducible representation of D_N) into one of two irreducible components.

Now we are ready to describe the algorithm of Kuperberg (2005). For simplicity, we assume that $N = 2^n$ is a power of 2. For such a dihedral group, it is actually sufficient to be able to determine the least significant bit of y since such an algorithm could be used recursively to

[11] The basic idea of the Ettinger-Høyer reduction is as follows. Suppose that $f: D_N \to S$ hides a subgroup $H = \langle (x,0), (y,1) \rangle$. Then we can consider the function f restricted to elements from the Abelian group $\mathbb{Z}/N\mathbb{Z} \times \{0\} \leq D_N$. This restricted function hides the subgroup $\langle (x,0) \rangle$, and since the restricted group is Abelian, we can find x efficiently using Algorithm 4. Now $\langle (x,0) \rangle \trianglelefteq D_N$ [since $(z,a)(x,0)(z,a)^{-1} = (z + (-1)^a x, a) \times (-(-1)^a z, a) = ((-1)^a x, 0) \in \mathbb{Z}/N\mathbb{Z} \times \{0\}$], so we can define the quotient group $D_N/\langle (x,0) \rangle$. But this is simply a dihedral group (of order N/x), and if we now define a function f' as f evaluated on some coset representative, it hides the subgroup $\langle (y,1) \rangle$.

determine all the bits of y.[12] Our strategy for doing this is to start with a large number of states and collect them into pairs $|\psi_p\rangle$, $|\psi_q\rangle$ that share many of their least significant bits, such that $|\psi_{p-q}\rangle$ is likely to have many of its least significant bits equal to 0. Trying to zero out all but the most significant bit in one shot would take exponentially long, so instead we proceed in stages, only trying to zero some of the least significant bits in each stage; this turns out to be an improvement. [This approach is similar to previous classical sieve algorithms for learning (Blum *et al.*, 2003) and lattice (Ajtai *et al.*, 2001) problems, as well as a subsequent classical algorithm for average case instances of subset sum (Flaxman and Przydatek, 2005).]

Algorithm 10 (*Kuperberg sieve*).

Input: Black-box function $f:D_{2^n}\to S$ hiding $\langle(y,1)\rangle \leqslant D_{2^n}$ for some $y\in\mathbb{Z}/2^n\mathbb{Z}$.

Problem: Determine the least significant bit of y.

(1) Prepare $\Theta(16^{\sqrt{n}})$ coset states of form (140), where each copy has $k\in\mathbb{Z}/2^n\mathbb{Z}$ chosen independently and uniformly at random.

(2) For each $j=0,1,\ldots,m-1$, where $m:=\lfloor\sqrt{n}\rfloor$, assume the current coset states have indices k with at least mj of the least significant bits equal to 0. Collect them into pairs $|\psi_p\rangle$, $|\psi_q\rangle$ that share at least m of the next least significant bits, discarding any qubits that cannot be paired. Create a state $|\psi_{p\pm q}\rangle$ from each pair and discard it if the $+$ sign occurs. Notice that the resulting states have at least $m(j+1)$ significant bits equal to 0.

(3) The remaining states are of the form $|\psi_0\rangle$ and $|\psi_{2^{n-1}}\rangle$. Measure one of the latter states in the $|\pm\rangle$ basis to determine the least significant bit of y.

Since this algorithm requires $2^{O(\sqrt{n})}$ initial queries and proceeds through $O(\sqrt{n})$ stages, each of which takes at most $2^{O(\sqrt{n})}$ steps, the overall running time is $2^{O(\sqrt{n})}$.

To show that the algorithm works, we need to prove that some qubits survive to the final stage of the process with non-negligible probability. We analyze a more general version of the algorithm to see why we should try to zero out $\sqrt{n}$ bits at a time, starting with $2^{O(\sqrt{n})}$ states.

Suppose we try to cancel m bits in each stage, so that there are n/m stages (not yet assuming any relationship between m and n), starting with 2^ℓ states. Each combination operation succeeds with probability $1/2$ and turns two states into one, so at each step we retain only about $1/4$ of the states that can be paired. Now when we pair states that allow us to cancel m bits, there can be at most 2^m unpaired states since that is the number of values of the m bits to be canceled. Thus if we ensure that there are at least 2×2^m states at each stage, we expect to retain at least a $1/8$ fraction of the states for the next stage. Since we begin with 2^ℓ states, we expect to have at least $2^{\ell-3j}$ states left after the jth stage. Thus, to have 2×2^m states remaining at the last stage of the algorithm, we require that $2^{\ell-3n/m}>2^{m+1}$ or $\ell>m+3n/m+1$. This is minimized by choosing $m\approx\sqrt{n}$, so $\ell\approx 4\sqrt{n}$ suffices.

This analysis is not quite correct because we do not obtain precisely a $1/8$ fraction of the paired states for use in the next stage. For most of the stages, we have many more than 2×2^m states, so nearly all of them can be paired, and the expected fraction remaining for the next stage is close to $1/4$. Of course, the precise fraction will experience statistical fluctuations. Since we are working with a large number of states, the deviations from the expected values are very small, and a more careful analysis (using the Chernoff bound) shows that the procedure succeeds with high probability. For a detailed argument, see Kuperberg (2005). Kuperberg also gives an improved algorithm that runs faster and that works for general N.

Note that this algorithm uses not only superpolynomial time but also superpolynomial space since all $2^{\Theta(\sqrt{n})}$ coset states are present at the start. However, by creating a smaller number of coset states at a time and combining them according to the solution of a subset sum problem, Regev (2004c) showed how to make the space requirement polynomial in n with only a slight increase in the running time.

Although Kuperberg's algorithm acts on pairs of coset states at a time, the overall algorithm effectively implements a highly entangled measurement on all $2^{\Theta(\sqrt{n})}$ registers since the procedure for producing $|\psi_{p\pm q}\rangle$ entangles the coset states $|\psi_p\rangle$ and $|\psi_q\rangle$. The same is true of Regev's polynomial-space variant.

It is natural to ask whether a similar sieve could be applied to other HSPs, such as in the symmetric group, for which highly entangled measurements are necessary. Alagic *et al.* (2007) adapted Kuperberg's approach to give a subexponential-time algorithm for the HSP in G^n, where G is a fixed non-Abelian group. (Note that the HSP in G^n can be much harder than solving n instances of the HSP in G since G^n has many subgroups that are not direct products of subgroups of G.) Bacon (2008) also showed that an algorithm for the Heisenberg HSP, similar to the one described in Sec. VII.G, can be derived using the Clebsch-Gordan transform over the Heisenberg group. It would be interesting to find further applications of the approach, especially ones that give new polynomial-time algorithms.

Unfortunately, this kind of sieve does not seem well suited to the symmetric group. In particular, Moore, Russell, and Sniady (2007) gave the following negative

[12]To see this, note that the group D_N contains two subgroups isomorphic to $D_{N/2}$, namely, $\{(2x,0),(2x,1):x\in\mathbb{Z}/(N/2)\mathbb{Z}\}$ and $\{(2x,0),(2x+1,1):x\in\mathbb{Z}/(N/2)\mathbb{Z}\}$. The hidden subgroup is a subgroup of the former if y has even parity and of the latter if y has odd parity. Thus, once we learn the parity of y, we restrict our attention to the appropriate $D_{N/2}$ subgroup. The elements of either $D_{N/2}$ subgroup can be represented using only $n-1$ bits and finding the least significant bit of the hidden reflection within this group corresponds to finding the second least significant bit of y in D_N. Continuing in this way, we learn all the bits of y with only n iterations of an algorithm for finding the least significant bit of the hidden reflection.

result for the HSP in $S_n \wr S_2$, where the hidden subgroup is promised to be either trivial or an involution. Consider any algorithm that works by combining pairs of hidden subgroup states to produce a new state in their Clebsch-Gordan decomposition and uses the sequence of measurement results to guess whether the hidden subgroup is trivial or nontrivial. Any such algorithm must use $2^{\Omega(\sqrt{n})}$ queries. Note that this lower bound is only slightly smaller than the best known classical algorithm for graph isomorphism, as discussed in Sec. VII.A.

G. Pretty good measurement

Another recent technique for the HSP is based on implementing the PGM on the hidden subgroup states. Recall from Sec. VII.E that for any group G the PGM applied to poly($\log|G|$) copies of ρ_H identifies H with high probability. Thus if we can efficiently implement the PGM on sufficiently many copies, we will have found an efficient algorithm for the HSP.

This approach was considered by Bacon *et al.* (2005, 2006) for certain semidirect product groups $A \rtimes \mathbb{Z}/p\mathbb{Z}$, where A is an Abelian group and p is prime. For these groups, the general HSP can be reduced to the HSP assuming that the hidden subgroup is chosen from a certain subset. Furthermore, the PGM turns out to be the optimal measurement for distinguishing the resulting hidden subgroup states in the sense that it maximizes the probability of correctly identifying the hidden subgroup assuming a uniform distribution over the subgroups under consideration [as can be proven using the characterization of optimal measurement by Holevo (1973) and Yuen *et al.* (1975)]. This generalizes the result of Ip (2003) that Shor's algorithm implements the optimal measurement for the Abelian HSP and suggests that, in general, optimal measurements may be good candidates for efficient quantum algorithms.

For general groups of the form $A \rtimes \mathbb{Z}/p\mathbb{Z}$, the PGM approach reveals a connection between the original hidden subgroup problem and a related average-case algebraic problem. Specifically, the PGM succeeds in distinguishing the hidden subgroup states exactly when the average-case problem is likely to have solutions and the PGM can be implemented efficiently by giving an efficient algorithm for solving the average-case problem (or more precisely for approximately quantum sampling from the set of solutions to the problem). Different HSPs correspond to different average-case problems of varying difficulty. For example, the dihedral HSP corresponds to the average-case subset sum problem (Bacon *et al.*, 2006), which appears to be hard to solve. But other average-case problems appearing in the approach are easier, leading to efficient algorithms. Certain instances of the Abelian HSP give rise to systems of linear equations. For the metacyclic HSPs solved by Moore, Rockmore, *et al.* (2007) (and indeed for some additional cases), the average-case problem is a discrete logarithm problem, which can be solved using Shor's algorithm as described in Sec. IV.B. And for the HSP in the Heisenberg group[13] $(\mathbb{Z}/p\mathbb{Z})^2 \rtimes \mathbb{Z}/p\mathbb{Z}$, and more generally in any semidirect product $(\mathbb{Z}/p\mathbb{Z})^r \rtimes \mathbb{Z}/p\mathbb{Z}$, the average-case problem is a problem of solving polynomial equations, which can be done efficiently using Gröbner basis techniques provided $r=O(1)$ (Bacon *et al.*, 2005).

Here we summarize the algorithm that results from applying the PGM to the HSP in the Heisenberg group since this case exemplifies the general approach. The Heisenberg group can be viewed as the semidirect product $(\mathbb{Z}/p\mathbb{Z})^2 \rtimes_\varphi \mathbb{Z}/p\mathbb{Z}$, where $\varphi: \mathbb{Z}/p\mathbb{Z} \to \mathrm{Aut}((\mathbb{Z}/p\mathbb{Z})^2)$ is defined by $\varphi(c)(a,b)=(a+bc,b)$. Equivalently, it is the group of lower triangular 3×3 matrices

$$\left\{ \begin{pmatrix} 1 & 0 & 0 \\ b & 1 & 0 \\ a & c & 1 \end{pmatrix} : a,b,c \in \mathbb{F}_p \right\} \tag{142}$$

over $\mathbb{F}_p$ or, alternatively, the group generated by generalized Pauli operators $X, Z \in \mathbb{C}^{p \times p}$ satisfying $X|x\rangle = |x+1 \bmod p\rangle$ and $Z|x\rangle = \omega_p^x |x\rangle$, with elements $\omega_p^a X^b Z^c$. With any of these descriptions, the group elements are of the form (a,b,c) with $a,b,c \in \mathbb{Z}/p\mathbb{Z}$, and the group law is

$$(a,b,c) \cdot (a',b',c') = (a+a'+b'c, b+b', c+c'). \tag{143}$$

Just as the dihedral HSP can be reduced to the problem of finding a hidden reflection (footnote 11) one can show that to solve the general HSP in the Heisenberg group, it is sufficient to be able to distinguish the following cyclic subgroups of order p:

$$H_{a,b} := \langle (a,b,1) \rangle = \{ (a,b,1)^j : j \in \mathbb{Z}/p\mathbb{Z} \}, \tag{144}$$

where $a,b \in \mathbb{Z}/p\mathbb{Z}$. A simple calculation shows that

$$(a,b,1)^x = \left(xa + \tbinom{x}{2} b, xb, x\right). \tag{145}$$

Furthermore, the cosets of any such subgroup can be represented by the p^2 elements $(\ell,m,0)$ for $\ell,m \in \mathbb{Z}/p\mathbb{Z}$. Thus the coset state [Eq. (112)] can be written as

$$|(\ell,m,0)H_{a,b}\rangle = \frac{1}{\sqrt{p}} \sum_{x \in \mathbb{Z}/p\mathbb{Z}} \left| xa + \tbinom{x}{2} b + \ell, xb + m, j \right\rangle. \tag{146}$$

Our goal is to determine the parameters $a,b \in \mathbb{Z}/p\mathbb{Z}$ using copies of this state with $\ell,m \in \mathbb{Z}/p\mathbb{Z}$ occurring uniformly at random.

At this point, we could perform weak Fourier sampling over the Heisenberg group without discarding any information. However, as for the dihedral group (Sec. VII.F), it is simpler to consider an Abelian Fourier transform instead of the full non-Abelian Fourier trans-

[13]The Heisenberg group is an example of an *extraspecial group*. Ivanyos *et al.* (2007) gave an efficient quantum algorithm for the HSP in any extraspecial group (see Sec. VIII.C for more details). This subsequent algorithm also makes use of the solution of a system of polynomial equations to implement an entangled measurement.

form. Using the representation theory of the Heisenberg group [see, e.g., Terras (1999)], one can show that this procedure is essentially equivalent to non-Abelian Fourier sampling.

Fourier transforming the first two registers over $(\mathbb{Z}/p\mathbb{Z})^2$, we obtain the state

$$\frac{1}{p^{3/2}} \sum_{x,s,t \in \mathbb{Z}/p\mathbb{Z}} \omega_p^{s[\ell+xa+\binom{x}{2}b]+t(m+xb)} |s,t,x\rangle. \tag{147}$$

Now suppose we measure the values s,t appearing in the first two registers. In fact this can be done without loss of information since the density matrix of the state (mixed over the uniformly random values of ℓ,m) is block-diagonal, with blocks labeled by s,t. Collecting the coefficients of the unknown parameters a,b, the resulting p-dimensional quantum state is

$$\widehat{|H_{a,b;s,t}\rangle} := \frac{1}{\sqrt{p}} \sum_{x \in \mathbb{Z}/p\mathbb{Z}} \omega_p^{a(sx)+b[s\binom{x}{2}+tx]} |x\rangle, \tag{148}$$

where the values $s,t \in \mathbb{Z}/p\mathbb{Z}$ are known and are obtained uniformly at random. We want to use samples of this state to determine $a,b \in \mathbb{Z}/p\mathbb{Z}$.

With only one copy of this state, there is insufficient information to recover the hidden subgroup: Holevo's theorem [see, e.g., Nielsen and Chuang (2000), Sec. 12.1] guarantees that a measurement on a p-dimensional quantum state can reliably communicate at most p different outcomes, yet there are p^2 possible values of $(a,b) \in (\mathbb{Z}/p\mathbb{Z})^2$. Thus we must use at least two copies.

By making a joint measurement on two copies of the state, we can recover the information about a,b that is encoded in a quadratic function in the phase. To see this, consider the state

$$\widehat{|H_{a,b;s,t}\rangle} \otimes \widehat{|H_{a,b;u,v}\rangle} = \frac{1}{p} \sum_{x,y \in \mathbb{Z}/p\mathbb{Z}} \omega_p^{\alpha a+\beta b} |x,y\rangle, \tag{149}$$

where

$$\alpha := sx + uy, \tag{150}$$

$$\beta := s\binom{x}{2} + tx + u\binom{y}{2} + vy, \tag{151}$$

and where we suppress the dependence of α, β on s,t,u,v,x,y for clarity. If we could replace $|x,y\rangle$ by $|\alpha,\beta\rangle$, then the resulting state would be simply the Fourier transform of $|a,b\rangle$, and an inverse Fourier transform would reveal the solution. To work toward this situation we compute the values of α,β in ancilla registers, giving the state

$$\frac{1}{p} \sum_{x,y \in \mathbb{Z}/p\mathbb{Z}} \omega_p^{\alpha a+\beta b} |x,y,\alpha,\beta\rangle, \tag{152}$$

and attempt to uncompute the first two registers.

For fixed values of $\alpha,\beta,s,t,u,v \in \mathbb{Z}/p\mathbb{Z}$, these two quadratic equations [Eqs. (150) and (151)] could have zero, one, or two solutions $x,y \in \mathbb{Z}/p\mathbb{Z}$. Thus we cannot hope to erase the first and second registers by a classical procedure conditioned on the values in the third and fourth registers (and the known values of s,t,u,v). However, it is possible to implement a quantum procedure to erase the first two registers by considering the full set of solutions,

$$S_{\alpha,\beta}^{s,t,u,v} := \{(x,y) \in (\mathbb{Z}/p\mathbb{Z})^2 : sx + uy = \alpha \text{ and } s\binom{x}{2} + tx + u\binom{y}{2} + vy = \beta\}. \tag{153}$$

The state Eq. (152) can be rewritten as

$$\frac{1}{p} \sum_{x,y \in \mathbb{Z}/p\mathbb{Z}} \omega_p^{\alpha a+\beta b} \sqrt{|S_{\alpha,\beta}^{s,t,u,v}|} |S_{\alpha,\beta}^{s,t,u,v}, \alpha,\beta\rangle. \tag{154}$$

Thus, if we could perform a unitary transformation satisfying

$$|S_{\alpha,\beta}^{s,t,u,v}\rangle \mapsto |\alpha,\beta\rangle \text{ for } |S_{\alpha,\beta}^{s,t,u,v}| \neq 0 \tag{155}$$

(and defined in any way consistent with unitarity for other values of α,β), we could erase the first two registers of Eq. (152),[14] producing the state

$$\frac{1}{p} \sum_{\alpha,\beta \in \mathbb{Z}/p\mathbb{Z}} \omega_p^{\alpha a+\beta b} \sqrt{|S_{\alpha,\beta}^{s,t,u,v}|} |\alpha,\beta\rangle. \tag{156}$$

The inverse of the transformation Eq. (155) is called *quantum sampling* because it produces a uniform superposition over the set of solutions, a natural quantum analog of *random sampling* from the solutions.

Since the system of Eqs. (150) and (151) consists of a pair of quadratic equations in two variables over $\mathbb{F}_p$, it has either zero, one, or two solutions $x,y \in \mathbb{F}_p$. For about half the cases, there are zero solutions; for about half the cases, there are two solutions; and for a vanishing fraction of the cases, there is only one solution. More explicitly, by a straightforward calculation, the solutions can be expressed in closed form as

$$x = \frac{\alpha s + sv - tu \pm \sqrt{\Delta}}{s(s+u)}, \tag{157}$$

$$y = \frac{\alpha u + tu - sv \mp \sqrt{\Delta}}{u(s+u)}, \tag{158}$$

where

$$\Delta := (2\beta s + \alpha s - \alpha^2 - 2\alpha t)(s+u)u + (\alpha u + tu - sv)^2. \tag{159}$$

Provided $su(s+u) \neq 0$, the number of solutions is completely determined by the value of Δ. If Δ is a nonzero square in $\mathbb{F}_p$, then there are two distinct solutions; if $\Delta = 0$ then there is only one solution; and if Δ is a nonsquare then there are no solutions. In any event, since we can efficiently compute an explicit list of solutions in each of these cases, we can efficiently perform the transformation [Eq. (155)].

[14] Note that we can apply the transformation [Eq. (155)] directly to the state [Eq. (149)]; there is no need to explicitly compute the values α,β in an ancilla register.

It remains to show that the state Eq. (156) can be used to recover a,b. This state is close to the Fourier transform of $|a,b\rangle$ provided the solutions are nearly uniformly distributed. Since the values of s,t,u,v are uniformly distributed over $\mathbb{F}_p$, it is easy to see that Δ is uniformly distributed over $\mathbb{F}_p$. This means that Δ is a square about half the time and is a nonsquare about half the time (with $\Delta=0$ occurring only with probability $1/p$). Thus there are two solutions about half the time and no solutions about half the time. This distribution of solutions is uniform enough for the procedure to work.

Applying the inverse quantum Fourier transform over $\mathbb{Z}/p\mathbb{Z}\times\mathbb{Z}/p\mathbb{Z}$, we obtain the state

$$\frac{1}{p^2}\sum_{\alpha,\beta,k,\ell\in\mathbb{Z}/p\mathbb{Z}}\omega_p^{\alpha(a-k)+\beta(b-\ell)}\sqrt{|S_{\alpha,\beta}^{s,t,u,v}|}|k,\ell\rangle. \tag{160}$$

Measuring this state, the probability of obtaining the outcome $k=a$ and $\ell=b$ for any particular values of s,t,u,v is

$$\frac{1}{p^4}\left(\sum_{\alpha,\beta\in\mathbb{Z}/p\mathbb{Z}}\sqrt{|S_{\alpha,\beta}^{s,t,u,v}|}\right)^2. \tag{161}$$

Since those values occur uniformly at random, the overall success probability of the algorithm is

$$\begin{aligned}&\frac{1}{p^8}\sum_{s,t,u,v\in\mathbb{Z}/p\mathbb{Z}}\left(\sum_{\alpha,\beta\in\mathbb{Z}/p\mathbb{Z}}\sqrt{|S_{\alpha,\beta}^{s,t,u,v}|}\right)^2\\&\quad\geqslant\frac{1}{p^{12}}\left(\sum_{s,t,u,v\in\mathbb{Z}/p\mathbb{Z}}\sum_{\alpha,\beta\in\mathbb{Z}/p\mathbb{Z}}\sqrt{|S_{\alpha,\beta}^{s,t,u,v}|}\right)^2\\&\quad\geqslant\frac{1}{p^{12}}\left(\sum_{\alpha,\beta\in\mathbb{Z}/p\mathbb{Z}}\frac{p^4}{2+o(1)}\sqrt{2}\right)^2=\frac{1}{2}-o(1),\end{aligned} \tag{162}$$

which shows that the algorithm succeeds with probability close to $1/2$.

In summary, the efficient quantum algorithm for the HSP in the Heisenberg group is as follows.

Algorithm 11 (Heisenberg HSP).

Input: Black-box function hiding $H_{a,b}$.

Problem: Determine the parameters a,b.

(1) Prepare two coset states, as in Eq. (146).

(2) Perform the QFT $F_{\mathbb{Z}/p\mathbb{Z}\times\mathbb{Z}/p\mathbb{Z}}$ on the first two registers of each coset state and measure those registers in the computational basis, giving Eq. (149).

(3) Perform the inverse quantum sampling transformation Eq. (155), giving Eq. (156).

(4) Perform the inverse QFT $F^\dagger_{\mathbb{Z}/p\mathbb{Z}\times\mathbb{Z}/p\mathbb{Z}}$, giving Eq. (160).

(5) Measure the resulting state in the computational basis, giving (a,b) with probability $1/2-o(1)$.

Because the transformation Eq. (155) acts jointly on the two registers, the algorithm described above effectively makes an entangled measurement on two copies of the hidden subgroup state. However, we do not know whether this is the only way to give an efficient algorithm for the HSP in the Heisenberg group. In particular, recall from Sec. VII.D that Fourier sampling in a random basis provides sufficient information to reconstruct the hidden subgroup (Radhakrishnan *et al.*, 2009). It would be interesting to know whether there is an *efficient* quantum algorithm using only the statistics of single-register measurements or if no such algorithm exists. It would also be interesting to find any group for which Fourier sampling does not suffice, even information theoretically, but for which there is an efficient quantum algorithm based on multiregister measurements.

The PGM approach outlined above can also be applied to certain state distinguishability problems that do not arise from HSPs. In particular, it can be applied to the generalized Abelian hidden shift problem discussed in Sec. VIII (for which the average-case problem is an integer program) (Childs and van Dam, 2007) and to hidden polynomial problems of the form (191), as discussed in Sec. IX (for which the average-case problem is again a system of polynomial equations) (Decker *et al.*, 2009).

VIII. HIDDEN SHIFT PROBLEM

The hidden shift problem (also known as the hidden translation problem) is a natural variant of the hidden subgroup problem. Its study has shed light on and, indeed, led to new algorithms for, the HSP. Furthermore, the hidden shift problem has applications that are of interest in their own right.

In the hidden shift problem, we are given two injective functions $f_0:G\to S$ and $f_1:G\to S$, with the promise that

$$f_0(g)=f_1(sg)\quad\text{for some } s\in G. \tag{163}$$

The goal of the problem is to find s, the hidden shift. In the non-Abelian hidden shift problem, as in the non-Abelian HSP, there is an arbitrary choice of left or right multiplication; here we again make the choice of left multiplication.

When G is Abelian, this problem is equivalent to the HSP in $G\rtimes_\varphi\mathbb{Z}/2\mathbb{Z}$ (sometimes called the G-dihedral group), where the homomorphism $\varphi:\mathbb{Z}/2\mathbb{Z}\to\mathrm{Aut}(G)$ is defined by $\varphi(0)(x)=x$ and $\varphi(1)(x)=x^{-1}$. In particular, the hidden shift problem in $\mathbb{Z}/N\mathbb{Z}$ is equivalent to the dihedral HSP. To see this, consider the function $f:G\rtimes\mathbb{Z}/2\mathbb{Z}\to S$ defined by $f(x,b):=f_b(x)$. This function hides the involution $\langle(s,1)\rangle$, so a solution of the HSP gives a solution of the hidden shift problem. Conversely, solving the HSP in $G\rtimes\mathbb{Z}/2\mathbb{Z}$ with the promise that H is an involution is sufficient to solve the HSP in general (footnote 11) so a solution of the hidden shift problem gives a solution of the HSP. While no polynomial-time quantum algorithm is known for the general Abelian hidden shift problem, Kuperberg's sieve (Algorithm 10) solves the problem in time $2^{O(\sqrt{\log|G|})}$, whereas a brute force approach takes $2^{\Omega(\log|G|)}$ steps.

When G is non-Abelian, the inversion map $x\mapsto x^{-1}$ is not a group automorphism, so we cannot even define a group $G\rtimes_\varphi\mathbb{Z}/2\mathbb{Z}$. However, the hidden shift problem in G is closely connected to an HSP, namely, in the

wreath product group $G \wr \mathbb{Z}/2\mathbb{Z} = (G \times G) \rtimes_{\tilde{\varphi}} \mathbb{Z}/2\mathbb{Z}$, where $\tilde{\varphi}(0)(x,y) = (x,y)$ and $\tilde{\varphi}(1)(x,y) = (y,x)$. The hidden shift problem in G reduces to the HSP in $G \wr \mathbb{Z}/2\mathbb{Z}$ with the hidden subgroup $\langle (s, s^{-1}, 1) \rangle$. Furthermore, the HSP in $G \wr \mathbb{Z}/2\mathbb{Z}$ with hidden subgroups of this form reduces to the hidden shift problem in $G \times G$. Thus, for families of groups in which $G \times G$ is contained in a larger group G' from the same family (such as for the symmetric group, where $S_n \times S_n \leq S_{2n}$) the hidden shift and hidden subgroup problems are essentially equivalent (Hallgren *et al.*, 2006). Moreover, by a similar argument to the one in Sec. VII.E, the quantum query complexity of the hidden shift problem in G is poly($\log|G|$) even when G is non-Abelian.

Testing isomorphism of rigid graphs can be cast as a hidden shift problem in the symmetric group. If we let $f(\pi,0) = \pi(\Gamma)$ and $f(\pi,1) = \pi(\Gamma')$, then the hidden shift is σ, where $\Gamma = \sigma(\Gamma')$. Despite the equivalence between hidden shift and hidden subgroup problems, the hidden shift problem in S_n is arguably a more natural setting for rigid graph isomorphism than the HSP since every possible hidden shift corresponds to a possible isomorphism between graphs, whereas the HSP must be restricted to certain subgroups (Childs and Wocjan, 2007).

In this section we describe quantum algorithms for various hidden shift problems. We begin by presenting a single-register measurement for the cyclic hidden shift problem (i.e., the dihedral HSP) that provides sufficient information to encode the hidden shift. While no efficient way of postprocessing this information is known, we explain how a similar approach leads to an efficient quantum algorithm for the hidden shift problem over $(\mathbb{Z}/p\mathbb{Z})^n$ with p a fixed prime. Since both of these problems are Abelian hidden shift problems, they could equally well be viewed as HSPs, but we discuss them here because the latter is an important ingredient of the *orbit coset* approach, which uses self-reducibility of a quantum version of the hidden shift problem to give efficient quantum algorithms for certain hidden subgroup and hidden shift problems. Then we describe an algorithm for the shifted Legendre symbol problem, a noninjective variant of the dihedral HSP that can be solved efficiently and that also leads to an efficient quantum algorithm for estimating Gauss sums. Finally, we describe a generalization of the hidden shift problem that interpolates to an Abelian HSP and that can be solved efficiently in some cases even when the original hidden shift problem cannot.

A. Abelian Fourier sampling for the dihedral HSP

Consider the HSP in the dihedral group $\mathbb{Z}/N\mathbb{Z} \rtimes \mathbb{Z}/2\mathbb{Z}$ with hidden subgroup $\langle (s,1) \rangle$ or, equivalently, the hidden shift problem in the cyclic group $\mathbb{Z}/N\mathbb{Z}$ with hidden shift s. Recall from Sec. VII.F [specifically Eq. (140)] that the standard method, followed by a measurement of the first register in the Fourier basis (over $\mathbb{Z}/N\mathbb{Z}$), produces the state $(|0\rangle + \omega_N^{sk}|1\rangle)/\sqrt{2}$ for some uniformly random measurement outcome $k \in \mathbb{Z}/N\mathbb{Z}$. Now suppose we measure this qubit in the basis of states $|\pm\rangle := (|0\rangle \pm |1\rangle)/\sqrt{2}$ (i.e., the Fourier basis over $\mathbb{Z}/2\mathbb{Z}$); then the outcome "+" occurs with probability $\cos^2(\pi sk/N)$. Thus, if we keep only those measured values of k for which the outcome of the second measurement is "+", we effectively sample from a distribution over $k \in \mathbb{Z}/N\mathbb{Z}$ with $\Pr(k) = 2\cos^2(\pi sk/N)/N$.

This procedure was proposed by Ettinger and Høyer (2000), who showed that $O(\log N)$ samples of the resulting distribution provide sufficient information to determine k with high probability. This single-register measurement is a much simpler procedure than either the Kuperberg sieve (Kuperberg, 2005) or the optimal measurement described by Bacon *et al.* (2006), both of which correspond to highly entangled measurements. However, we are left with the problem of postprocessing the measurement results to infer the value of s, for which no efficient procedure is known.

B. Finding hidden shifts in $(\mathbb{Z}/p\mathbb{Z})^n$

A similar approach can be applied to the hidden shift problem in the elementary Abelian p-group $(\mathbb{Z}/p\mathbb{Z})^n$ with p a fixed prime, but in this case the postprocessing can be carried out efficiently. This result is an important building block in an efficient quantum algorithm for the hidden shift and hidden subgroup problems in certain families of solvable groups (Friedl *et al.*, 2003), as discussed in the next section.

Consider the hidden shift problem in $(\mathbb{Z}/p\mathbb{Z})^n$ with hidden shift s. Applying the standard method, we obtain the hidden shift state

$$\frac{1}{\sqrt{2}}(|z,0\rangle + |z+s,1\rangle) \tag{164}$$

for some unknown $z \in (\mathbb{Z}/p\mathbb{Z})^n$ chosen uniformly at random. Now suppose that, as in the measurement for the dihedral group described above, we perform Abelian Fourier sampling on this state. In other words, we Fourier transform the first register over $(\mathbb{Z}/p\mathbb{Z})^n$ and the second over $\mathbb{Z}/2\mathbb{Z}$; this gives

$$\frac{1}{2\sqrt{p^n}} \sum_{y \in (\mathbb{Z}/p\mathbb{Z})^n} \sum_{b \in \mathbb{Z}/2\mathbb{Z}} [\omega_p^{y \cdot z} + \omega_p^{y \cdot (z+s)}(-1)^b] |y,b\rangle. \tag{165}$$

Finally, suppose we measure this state in the computational basis. A straightforward calculation shows that we obtain the outcome $(y,0)$ with probability $\cos^2(\pi y \cdot s/p)/p^n$ and the outcome $(y,1)$ with probability $\sin^2(\pi y \cdot s/p)/p^n$. Thus, conditioned on observing 1 in the second register, we see y in the first register with probability

$$\Pr(y) = \frac{2}{p^n} \sin^2 \frac{\pi y \cdot s}{p}. \tag{166}$$

In particular, note that there is zero probability of seeing any $y \in (\mathbb{Z}/p\mathbb{Z})^n$ such that $y \cdot s = 0 \bmod p$: we see only points that are not orthogonal to the hidden shift. [This may be contrasted with the HSP in $(\mathbb{Z}/p\mathbb{Z})^n$ with hidden

subgroup $\langle s\rangle$, in which Fourier sampling only gives points $x \in (\mathbb{Z}/p\mathbb{Z})^n$ with $x\cdot s=0$.]

We now argue that $O(n)$ samples from this distribution are information-theoretically sufficient to determine the hidden shift s. Since we only observe points y that are not orthogonal to s, the observation of y allows us to eliminate the hyperplane $y\cdot s=0$ of possible values of s. With enough samples, we can eliminate all possible candidate values of s except the true value (and scalar multiples thereof).

For simplicity, suppose we sample uniformly from all $y \in (\mathbb{Z}/p\mathbb{Z})^n$ satisfying $y\cdot s\neq 0$ for the unknown s. While the true distribution Eq. (166) is not uniform, it is not far from uniform, so the argument given here can easily be modified to work for the true distribution. Consider some fixed candidate value s' with $s'\neq \alpha s$ for any $\alpha \in \mathbb{Z}/p\mathbb{Z}$. If y were sampled uniformly at random, then s' would be eliminated with probability $1/p$. Sampling uniformly from the subset of points y satisfying $y\cdot s\neq 0$ only raises the probability of eliminating s', so a randomly sampled y eliminates s' with probability at least $1/p$. Thus after $O(n)$ samples, the probability of not eliminating s' is exponentially small and, by a union bound, the probability of any such s' not being eliminated is upper bounded by a constant.

Unfortunately, given $k=\Theta(n)$ samples $y_1,\ldots,y_k$, we do not know how to efficiently determine s. We want to solve the system of inequations $y_1\cdot s\neq 0,\ldots,y_k\cdot s\neq 0$ for $s \in (\mathbb{Z}/p\mathbb{Z})^n$. Using Fermat's little theorem, which says that $a^{p-1}=1$ for any $a \in \mathbb{Z}/p\mathbb{Z}$ with $a\neq 0$, we can rewrite these inequations as a system of polynomial equations $(y_1\cdot s)^{p-1}=\cdots=(y_k\cdot s)^{p-1}=1$. However, the problem of solving polynomial equations over a finite field is NP-hard, so we cannot hope to solve for s quickly using generic methods.

This problem has been circumvented by Friedl *et al.* (2003) and Ivanyos (2008) using the idea of linearization. If we treat each product of $p-1$ components of $s \in (\mathbb{Z}/p\mathbb{Z})^n$ as a separate variable, then we can view $(y\cdot s)^{p-1}=1$ as a linear equation over a vector space of dimension $\binom{n+p-2}{p-1}$ (the number of ways of choosing $p-1$ items from n items, with replacement and without regard for ordering). Since this method treats variables as independent that are in fact highly dependent, it requires more samples to obtain a unique solution. Nevertheless, Friedl *et al.* (2003) showed that $O(n^{p-1})$ samples suffice. Since this method only involves linear equations and the number of equations remains poly(n) [recall the assumption that $p=O(1)$], the resulting algorithm is efficient.

A similar approach works for the hidden shift problem in $(\mathbb{Z}/p^k\mathbb{Z})^n$, where p^k is any fixed prime power (Friedl *et al.*, 2003; Ivanyos, 2008). However, no efficient algorithm is known for the case of $(\mathbb{Z}/m\mathbb{Z})^n$ with m not a prime power, even in the smallest case, $m=6$.

C. Self-reducibility, quantum hiding, and the orbit coset problem

By combining the result of the previous section with a *self-reducible* variant of the hidden shift problem, Friedl *et al.* (2003) also gave an efficient quantum algorithm for the HSP and hidden shift problem in a large family of solvable groups. The idea of self-reducibility is as follows. Suppose we could reduce the HSP in G to the HSP in subgroups of G and apply such a reduction recursively until the remaining groups are either simple enough that the HSP can be solved by some known method or small enough that it can be solved by brute force. For example, it would be useful if we could reduce the HSP in G to the HSP in N and G/N, where $N\triangleleft G$ is a proper normal subgroup of G. No approach of this kind has been directly applied to the HSP or the hidden shift problem, but this self-reducibility concept has proved fruitful for a quantum generalization of the hidden shift problem called the *orbit coset problem*.

Recall that in the standard method for the HSP, we prepare the uniform superposition $|G\rangle$, query a black-box function $f:G\to S$ satisfying Eq. (108), and discard the resulting function value, producing a uniformly random coset state $|xH\rangle$. More generally, suppose we have some black-box isometry F satisfying

$$F|x\rangle = |x\rangle \otimes |\phi_x\rangle \tag{167}$$

for some set of quantum states $\{|\phi_x\rangle : x \in G\}$ satisfying

$$\langle \phi_x|\phi_y\rangle = \begin{cases} 1 & \text{for } x^{-1}y \in H \\ 0 & \text{otherwise.} \end{cases} \tag{168}$$

By analogy to Eq. (108), we say that F is a *quantum hiding function* for H in G. Querying the quantum black box F on the uniform superposition $|G\rangle$ and discarding the second register has the same effect as the standard method: the result is a uniformly random coset state $|xH\rangle$. But the possibility of using quantum superpositions for the states $|\phi_x\rangle$ offers more freedom when constructing reductions.

One way to produce quantum hiding states $\{|\phi_x\rangle : x \in G\}$ is as follows. Let Φ be an orthonormal set of quantum states and let $\alpha: G\times\Phi\to\Phi$ be a (left) action of G on Φ. For some fixed $|\phi\rangle \in \Phi$, define $|\phi_x\rangle := \alpha(x)(|\phi\rangle)$. Then the isometry Eq. (167) is a quantum hiding function for the *stabilizer* of $|\phi\rangle$, the subgroup $\mathrm{stab}(|\phi\rangle) := \{x \in G : \alpha(x)(|\phi\rangle)=|\phi\rangle\} \leq G$. Fixing G, Φ, and α, the stabilizer problem[15] asks us to find a generating set for $\mathrm{stab}(|\phi\rangle)$ given (some number of copies of) the state $|\phi\rangle$.

In the same sense that the stabilizer problem can be viewed as an HSP with a quantum hiding function, the orbit coset problem is analogous to the hidden shift problem. The orbit coset of $|\phi_0\rangle, |\phi_1\rangle \in \Phi$ is the set $\{x \in G : \alpha(x)(|\phi_1\rangle)=|\phi_0\rangle\}$; it is either empty or a left coset

[15] Kitaev (1995) gave an efficient algorithm for the stabilizer problem in the case where G is Abelian and the hiding function is classical, prefiguring the hidden subgroup framework.

of stab($|\phi_1\rangle$) (or, equivalently, a right coset of $|\phi_0\rangle$). In the orbit coset problem (OCP), we are given (some number of copies of) $|\phi_0\rangle, |\phi_1\rangle \in \Phi$. The goal is to decide whether their orbit coset is nonempty and, if so, to find both a generating set for stab($|\phi_1\rangle$) and an element $x \in G$ such that $\alpha(x)(|\phi_0\rangle) = |\phi_1\rangle$.

It can be shown that for any group G and any solvable normal subgroup $N \triangleleft G$, the OCP in G reduces to the OCP in G/N and subgroups of N (Friedl *et al.*, 2003). While the details are beyond the scope of this article, the reduction is based on a method for creating a uniform superposition over the orbit of a state $|\phi\rangle$ under the action α, building on a technique introduced by Watrous in his algorithms for solvable groups (Sec. IV.G). By combining this with the efficient quantum algorithm for the hidden shift problem in $(\mathbb{Z}/p\mathbb{Z})^n$ discussed in Sec. VIII.B [which can be straightforwardly adapted to an efficient algorithm for orbit coset in $(\mathbb{Z}/p\mathbb{Z})^n$], Friedl *et al.* (2003) obtained an efficient quantum algorithm for the hidden shift problem in smoothly solvable groups and for the HSP in solvable groups with a smoothly solvable commutator subgroup.

Recently, Ivanyos, Sanselme, and Santha have given algorithms for the HSP in extraspecial groups (Ivanyos *et al.*, 2007) and groups of nilpotency class at most 2 (Ivanyos *et al.*, 2008). These algorithms use the concept of a quantum hiding function introduced above to reduce the problem to an Abelian HSP. It would be interesting to develop further applications of quantum hiding functions to the HSP, hidden shift, and related problems.

D. Shifted Legendre symbol and Gauss sums

While no efficient quantum algorithm is known for the cyclic hidden shift problem (i.e., the dihedral HSP) for a general function $f_0: \mathbb{Z}/N\mathbb{Z} \to S$, the problem can be more tractable given a hiding function of a particular form. As a simple example, the hidden shift problem with the identity function $f_0(x) = x$ is trivial; but this case is uninteresting as the problem can be solved equally well with a classical or quantum computer. However, more interesting examples can be constructed if we drop the requirement that f_0 be injective.[16] For example, the Legendre symbol χ provides an example of a function with an efficient quantum algorithm but no known efficient classical algorithm.

1. Shifted Legendre symbol problem

For a finite field $\mathbb{F}_p$ with p an odd prime, the value $\chi(x)$ of the *Legendre symbol* $\chi: \mathbb{F}_p \to \{-1, 0, +1\}$ depends on whether x is zero, a nonzero square (i.e., a quadratic residue), or a nonsquare (i.e., a quadratic nonresidue) in $\mathbb{F}_p$. It is defined by

$$\chi(x) = \begin{cases} 0 & \text{if } x = 0 \\ +1 & \text{if } \exists\, y \neq 0,\ x = y^2 \\ -1 & \text{otherwise.} \end{cases} \tag{169}$$

For example, in $\mathbb{F}_5$ we have the values

x	0	1	2	3	4
$\chi(x)$	0	+1	−1	−1	+1

The Legendre symbol is a multiplicative character, as it is easy to verify that $\chi(xy) = \chi(x)\chi(y)$ for all $x, y \in \mathbb{F}_p$. This fact can be used to show that $\Sigma_{x \in \mathbb{F}_p} \chi(x) = 0$. The identity

$$\chi(x) = x^{(p-1)/2} \bmod p \tag{170}$$

shows that repeated squaring mod p can be used to compute the value $\chi(x)$ in time poly(log p).

In the *shifted Legendre symbol problem* over $\mathbb{F}_p$, we define the functions $f_0(x) := \chi(x)$ and $f_1(x) := \chi(x+s)$ for all $s \in \mathbb{F}_p$; the task is to determine the hidden shift s given a black-box implementation of the function f_1. We emphasize that although the functions f_0, f_1 are not injective, this can nevertheless be viewed as (a relaxed version of) a hidden shift problem. The ability to efficiently solve this particular hidden shift problem quantum mechanically stems from properties of multiplicative functions under the (additive) Fourier transform.

No efficient classical algorithm for the shifted Legendre symbol problem is known. Although one can show that $O(\log p)$ random queries to the function $\chi(x+s)$ are sufficient to obtain enough information to determine s (van Dam, 2002), it is not clear how to do so efficiently. In fact, the Legendre sequence $\chi(x), \chi(x+1), \ldots$ has been proposed as a pseudorandom function with potential cryptographic applications (Damgård, 1990).

The following quantum algorithm efficiently solves the shifted Legendre symbol problem (van Dam *et al.*, 2006).

Algorithm 12 (Shifted Legendre symbol).

Input: Black-box function $\chi(x+s)$ for some unknown $s \in \mathbb{F}_p$.

Problem: Determine the hidden shift s.

(1) Prepare the uniform superposition $|\mathbb{F}_p\rangle$ and query the function in an ancilla register, giving the state

$$\frac{1}{\sqrt{p}} \sum_{x \in \mathbb{F}_p} |x, \chi(x+s)\rangle. \tag{171}$$

(2) Measure whether the second register is in the state $|0\rangle$. If it is, the first register is left in the state $|-s\rangle$, and measuring it determines s. Otherwise, we are left with the state

$$\frac{1}{\sqrt{p-1}} \sum_{x \in \mathbb{F}_p \setminus \{-s\}} |x, \chi(x+s)\rangle, \tag{172}$$

and we continue.

[16] Dropping this restriction, the quantum query complexity of the hidden shift problem may no longer be polynomial; for example, the hidden shift problem with $f_0(x) = \delta_{x,0}$ is equivalent to unstructured search, which has quantum query complexity $\Omega(\sqrt{N})$ (Bennett *et al.*, 1997).

(3) Apply the unitary operation $|x,b\rangle \mapsto (-1)^b|x,b\rangle$ and uncompute the shifted Legendre symbol, giving the state

$$\frac{1}{\sqrt{p-1}} \sum_{x\in\mathbb{F}_p} \chi(x+s)|x\rangle. \tag{173}$$

(4) Apply the Fourier transform over $\mathbb{F}_p$, yielding

$$\frac{1}{\sqrt{p-1}} \sum_{y\in\mathbb{F}_p} \hat{\chi}(y)\omega_p^{-sy}|y\rangle, \tag{174}$$

where $\hat{\chi}:\mathbb{F}_p\to\mathbb{C}$ is the normalized Fourier transform of χ [a normalized Gauss sum; cf. Eq. (180)], namely,

$$\hat{\chi}(y) := \frac{1}{\sqrt{p}} \sum_{x\in\mathbb{F}_p} \chi(x)\omega_p^{xy}. \tag{175}$$

Note that $\hat{\chi}(0)=0$ and $|\hat{\chi}(y)|=1$ for $y\in\mathbb{F}_p^\times$.

(5) The equality

$$\hat{\chi}(y) = \frac{1}{\sqrt{p}} \sum_{x\in\mathbb{F}_p} \chi(xy^{-1})\omega_p^{x} = \chi(y)\hat{\chi}(1) \tag{176}$$

shows that the state [Eq. (174)] is in fact a uniformly weighted superposition of the elements of $\mathbb{F}_p$, where the state $|y\rangle$ has a phase proportional to $\chi(y)\omega_p^{-sy}$. Thus we correct the relative phases by the operation $|y\rangle\mapsto\chi(y)|y\rangle$ for all $y\in\mathbb{F}_p^\times$, giving the state

$$\frac{\hat{\chi}(1)}{\sqrt{p-1}} \sum_{y\in\mathbb{F}_p^\times} \omega_p^{-sy}|y\rangle. \tag{177}$$

(6) Perform the Fourier transform over $\mathbb{F}_p$ and measure in the computational basis, giving s with probability $1-O(1/p)$.

It is easy to see that the above algorithm solves the shifted Legendre symbol problem not only over a prime field $\mathbb{F}_p$ but over any finite field $\mathbb{F}_q$. To verify this, we need only compute the Fourier transform of the quadratic character $\chi:\mathbb{F}_q\to\{-1,0,+1\}$, namely,

$$\begin{aligned}\hat{\chi}(y) &:= \frac{1}{\sqrt{q}} \sum_{x\in\mathbb{F}_q} \chi(x)\omega_p^{\mathrm{Tr}(xy)} = \frac{1}{\sqrt{q}} \sum_{x\in\mathbb{F}_q} \chi(xy^{-1})\omega_p^{\mathrm{Tr}(x)} \\ &= \chi(y)\hat{\chi}(1)\end{aligned} \tag{178}$$

(recall the definition of the Fourier transform over $\mathbb{F}_q$ in Sec. III.D). Indeed, the solution can be generalized to any shifted multiplicative character of $\mathbb{F}_q$ (van Dam *et al.*, 2006) and to any function over $\mathbb{F}_p$ that hides a multiplicative subgroup of polylogarithmic index (Moore, Rockmore, *et al.*, 2007).

For the ring $\mathbb{Z}/N\mathbb{Z}$ with $N=p_1^{r_1}\times\cdots\times p_k^{r_k}$ odd, the generalization of the Legendre symbol is called the *Jacobi symbol* $(\cdot/N):\mathbb{Z}/N\mathbb{Z}\to\{-1,0,+1\}$. It is defined as the product

$$\left(\frac{x}{N}\right) = \left(\frac{x}{p_1}\right)^{r_1}\cdots\left(\frac{x}{p_k}\right)^{r_k}, \tag{179}$$

where $(x/p) := \chi(x)$ is an alternative notation for the Legendre symbol that makes the field size explicit. This is again a multiplicative character, although its values need not indicate squares mod N [for example, $(2/15)=(2/3)(2/5)=(-1)^2=1$, while 2 is not a square mod 15]. Analogous to the shifted Legendre symbol problem, one can define a shifted Jacobi symbol problem over $\mathbb{Z}/N\mathbb{Z}$, which also has an efficient quantum algorithm (van Dam *et al.*, 2006).

2. Estimating Gauss sums

In the above solution to the shifted Legendre symbol problem, we encountered the Fourier transform of the multiplicative character χ, which is a *Gauss sum*. This naturally leads to a quantum algorithm for approximating Gauss sums (van Dam and Seroussi, 2002).

For a finite field $\mathbb{F}_q$, a nontrivial multiplicative character $\chi:\mathbb{F}_q\to\mathbb{C}$, and a nontrivial additive character $\psi:\mathbb{F}_q\to\mathbb{C}$, the Gauss sum is defined as the inner product between these two characters:

$$G(\chi,\psi) := \sum_{x\in\mathbb{F}_q} \chi(x)\psi(x). \tag{180}$$

It is not hard to show that any Gauss sum has norm $|G(\chi,\psi)|=\sqrt{q}$, so to learn the value of a Gauss sum it suffices to determine the phase $\phi\in[0,2\pi)$ of $G(\chi,\psi)=\sqrt{q}\cdot e^{i\phi}$.

There are $q-1$ distinct multiplicative characters $\chi_a:\mathbb{F}_q\to\mathbb{C}$ indexed by $a\in\mathbb{Z}/(q-1)\mathbb{Z}$. For a fixed multiplicative generator g of $\mathbb{F}_q^\times$, we have $\chi_a(g^j) := \omega_{q-1}^{aj}$ for all $j\in\mathbb{Z}$ and $\chi_a(0) := 0$. The $q-2$ nontrivial characters are those with $a\neq 0$. As the discrete logarithm $\log_g(g^j)=j \bmod q-1$ can be calculated efficiently with a quantum computer (Sec. IV.B), we can efficiently induce the phase $\chi_a(g^j)$ by subtracting the value $aj \bmod q-1$ from the state $|\hat{1}\rangle$, giving

$$\begin{aligned}|g^j\rangle\otimes|\hat{1}\rangle &= |g^j\rangle\otimes\frac{1}{\sqrt{q-1}}\sum_{y\in\mathbb{Z}/(q-1)\mathbb{Z}}\omega_{q-1}^{y}|y\rangle \\ &\mapsto |g^j\rangle\otimes\frac{1}{\sqrt{q-1}}\sum_{y\in\mathbb{Z}/(q-1)\mathbb{Z}}\omega_{q-1}^{y}|y-aj\rangle \\ &= |g^j\rangle\otimes\frac{\omega_{q-1}^{aj}}{\sqrt{q-1}}\sum_{y\in\mathbb{Z}/(q-1)\mathbb{Z}}\omega_{q-1}^{y}|y\rangle \\ &= \chi_a(g^j)|g^j\rangle\otimes|\hat{1}\rangle;\end{aligned} \tag{181}$$

this is sometimes referred to as the *phase kickback trick*.

The q additive characters $\psi_b:\mathbb{F}_q\to\mathbb{C}$ indexed by $b\in\mathbb{F}_q$ are defined as $\psi_b(x) := \omega_p^{\mathrm{Tr}(bx)}$ for all $x\in\mathbb{F}_q$. The character ψ_0 is trivial and all $b\neq 0$ give nontrivial characters.

With these definitions in place, the Gauss sum estimation algorithm is as follows.

Algorithm 13 (*Gauss sum estimation*).

Input: A finite field $\mathbb{F}_q$, a nontrivial multiplicative character χ_a [where $a\in(\mathbb{Z}/(q-1)\mathbb{Z})^\times$] and a nontrivial additive character ψ_b (where $b\in\mathbb{F}_q^\times$).

Problem: Approximate within precision $\delta>0$ the angle $\phi\in[0,2\pi)$ such that $G(\chi_a,\psi_b)=\sqrt{q}\cdot e^{i\phi}$.

Perform phase estimation (Sec. III.C) with precision δ on the following single-qubit unitary operation [which requires applying the operation $O(1/\delta)$ times], inputting its eigenstate $|1\rangle$ of eigenvalue $e^{i\phi}$.

(1) For an arbitrary input state $\alpha|0\rangle+\beta|1\rangle$, prepare the state $|\mathbb{F}_q^\times\rangle$ in an ancilla register.

(2) Using the phase kickback trick described in Eq. (181), transform the state to

$$\frac{1}{\sqrt{q-1}}\left(\alpha|0\rangle\otimes\sum_{x\in\mathbb{F}_q}\chi_a^*(x)|x\rangle+\beta|1\rangle\otimes\sum_{x\in\mathbb{F}_q}\chi_a(x)|x\rangle\right). \tag{182}$$

(3) Conditional on the qubit being in the state $|1\rangle$, multiply the ancilla register by b and apply the Fourier transform over $\mathbb{F}_q$, yielding the state

$$(\alpha|0\rangle+\hat{\chi}_a(b)\beta|1\rangle)\otimes\frac{1}{\sqrt{q-1}}\sum_{x\in\mathbb{F}_q}\chi_a^*(x)|x\rangle, \tag{183}$$

where

$$\hat{\chi}_a(b):=\frac{G(\chi_a,\psi_b)}{\sqrt{q}}=e^{i\phi}. \tag{184}$$

(4) Apply the phase rotation $|x\rangle\mapsto\chi_a(x)|x\rangle$ to the ancilla register, returning it to its original state and giving

$$(\alpha|0\rangle+\hat{\chi}_a(b)\beta|1\rangle)\otimes|\mathbb{F}_q^\times\rangle. \tag{185}$$

Discarding the ancilla register, notice that the above steps effectively implement the conditional phase shift $|0\rangle\mapsto|0\rangle$, $|1\rangle\mapsto e^{i\phi}|1\rangle$.

The above quantum algorithm has running time polynomial in $\log q$ and $1/\delta$, whereas classical sampling over the q values $\chi_a(x)\psi_b(x)$ requires poly$(\sqrt{q}/\delta)$ samples to achieve the same quality of approximation.

Both additive and multiplicative characters can be defined over the ring $\mathbb{Z}/N\mathbb{Z}$, and there are corresponding Gauss sums

$$G(\chi_a,\psi_b)=\sum_{x\in\mathbb{Z}/N\mathbb{Z}}\chi_a(x)\psi_b(x), \tag{186}$$

with $\chi_a(xy)=\chi_a(x)\chi_a(y)$ and $\psi_b(x)=\omega_N bx$ for all $x,y\in\mathbb{Z}/N\mathbb{Z}$ [see Berndt *et al.* (1998)]. Such Gauss sums over finite rings can be approximated by a quantum computer as well, using the above algorithm in a relatively straightforward way.

As Gauss sums occur frequently in the calculation of the number of points on hypersurfaces over finite fields [see, e.g., Ireland and Rosen (1990)] these same quantum algorithms can be used to approximately count such points with an accuracy that does not seem achievable classically (van Dam, 2004).

E. Generalized hidden shift problem

Pólya has advised that "if there is a problem you can't solve, then there is an easier problem you can solve: find it" (Pólya, 1945). In that spirit, we conclude our discussion of the hidden shift problem by describing a generalization that offers more ways to obtain information about the hidden shift. At least in the case of cyclic groups, this problem indeed turns out to be easier than the original hidden shift problem.

In the *M-generalized hidden shift problem* for the group G, we are given a hiding function $f:\{0,\ldots,M-1\}\times G\to S$ satisfying two conditions: for any fixed $j\in\{0,\ldots,M-1\}$, $f(j,x)$ is an injective function of $x\in G$ and for each $j\in\{0,\ldots,M-2\}$, $f(j+1,x)=f(j,sx)$. For $M=2$, this problem is equivalent to the usual hidden shift problem, since the hiding functions f_0,f_1 can be obtained as $f_j(x)=f(j,x)$. However, the M-generalized hidden shift problem appears to become easier for larger M; it trivially reduces to the M'-generalized hidden shift problem with $M'<M$, but larger values of M provide new ways to query the hiding function. Note that if $s^M=1$, then the M-generalized hidden shift problem is equivalent to the HSP in $\mathbb{Z}/M\mathbb{Z}\times G$ with the cyclic hidden subgroup $\langle(1,s)\rangle$. In general, the M-generalized hidden shift problem in G reduces to the HSP in $G\wr\mathbb{Z}/M\mathbb{Z}$ (Fenner and Zhang, 2008), but note that this reduction is only efficient for $M=$poly$(\log|G|)$.

The Abelian generalized hidden shift problem could potentially be applied to solve lattice problems. Recall from Sec. VII.A that the poly(n)-unique shortest lattice vector problem efficiently reduces to the standard approach to the dihedral HSP. In fact the same holds for the M-generalized hidden shift problem in $\mathbb{Z}/N\mathbb{Z}$, provided $M=$poly$(\log N)$.

While no efficient algorithm is known for the case where $M=$poly$(\log N)$, efficient algorithms do exist for larger values of M. Note that the N-generalized hidden shift problem in $\mathbb{Z}/N\mathbb{Z}$ is an HSP in $\mathbb{Z}/N\mathbb{Z}\times\mathbb{Z}/N\mathbb{Z}$, which can be solved by Abelian Fourier sampling. Essentially the same strategy works provided $M=\Omega(N)$, but fails for sublinear values of M. However, there is another quantum algorithm that is efficient provided $M\geqslant N^\epsilon$ for some fixed $\epsilon>0$ (Childs and van Dam, 2007) based on the pretty good measurement techniques discussed in Sec. VII.G. For the M-generalized hidden shift problem in $\mathbb{Z}/N\mathbb{Z}$, implementing the PGM reduces to an integer programming problem in $d=\log N/\log M$ dimensions, which can be solved efficiently for $d=O(1)$ (Lenstra, 1983).

It would also be interesting to consider the generalized hidden shift problem in non-Abelian groups. For example, a solution of this problem for the symmetric group could be used to solve the *M-generalized graph isomorphism problem*, in which we are given M rigid n-vertex graphs $\Gamma_0,\Gamma_1,\ldots,\Gamma_{M-1}$ that are all either nonisomorphic or sequentially isomorphic with a fixed isomorphism $\pi\in S_n$, namely, $\Gamma_{j+1}=\pi(\Gamma_j)$ for $j=0,1,\ldots,M-2$. For large M, this problem might seem considerably easier than graph isomorphism, yet no efficient algorithms for the corresponding generalized hidden shift problem are known. Indeed, very little is known about the non-Abelian generalized hidden shift problem in general.

IX. HIDDEN NONLINEAR STRUCTURES

The non-Abelian hidden subgroup problem (Sec. VII) was originally introduced with the hope of generalizing the success of Shor's algorithm. As we have seen, these efforts have so far met with only limited success: while polynomial-time quantum algorithms are known for the HSP in some non-Abelian groups, the cases with significant applications (namely, the dihedral and symmetric groups) remain largely unresolved. Thus there have been several attempts to generalize the Abelian HSP in other ways. The hidden shift problem (Sec. VIII) represents one such attempt. In this section we discuss a more radical departure from the HSP, a class of problems aimed at finding *hidden nonlinear structures*.

We return our attention to the Abelian HSP and, more specifically, to the hidden subgroup problem in the additive group of the d-dimensional vector space $\mathbb{F}_q^d$ (where $\mathbb{F}_q$ denotes the finite field with q elements). Then we can view the HSP as a problem of identifying a hidden linear structure: the subgroups of the additive group $\mathbb{F}_q^d$ are precisely its linear subspaces, and their cosets are parallel affine subspaces or *flats* [cf. step (4) of Algorithm 3]. Thus in this HSP we are given a function that is constant on sets of points specified by linear equations, and the goal is to recover certain parameters of those equations. It is natural to consider replacing the linear function by a polynomial of higher degree. Here we describe three such hidden nonlinear structure problems: the hidden polynomial problem, shifted subset problems, and polynomial Legendre symbol problems.

A. The hidden polynomial problem

Perhaps the most straightforward nonlinear generalization of the Abelian HSP is the *hidden polynomial problem* (Childs *et al.*, 2007). In this problem, the hidden object is a polynomial $h(x) \in \mathbb{F}_q[x_1, \ldots, x_d]$. Generalizing Eq. (38), we say that a black-box function $f: \mathbb{F}_q^d \to S$ (for some finite set S) hides the polynomial $h(x)$ if

$$f(x) = f(x') \text{ if and only if } h(x) = h(x') \tag{187}$$

for all $x, x' \in \mathbb{F}_q^d$. In other words, the function f is constant on the *level sets*

$$L_y^h := h^{-1}(y) = \{x \in \mathbb{F}_q^d : h(x) = y\} \tag{188}$$

and distinct on different level sets. The hidden polynomial problem is to determine $h(x)$ up to differences that do not affect its level sets (i.e., up to an overall additive or multiplicative constant).

Note that the polynomial $h(x)$ trivially hides itself. But just as there is no *a priori* relationship between function values and cosets in the general HSP, we prefer to assume that the association of function values to level sets is arbitrary. Indeed, if we were promised that $f(x) = h(x)$, even a classical computer could solve the hidden polynomial problem efficiently. But with no promise on how the level sets are mapped to function values, it is not hard to show that the classical randomized query complexity of the hidden polynomial problem is exponential in $d \log q$ (Childs *et al.*, 2007) by a similar argument as for the Abelian HSP (Simon, 1997).

With a quantum computer, we can approach the hidden polynomial problem by closely following the standard method for the HSP (Sec. VII.B). Querying the function f on the uniform superposition $|\mathbb{F}_q^d\rangle$ and discarding the resulting function value, one is left with the state $|L_y^h\rangle$ with probability $|L_y^h|/q^d$. Equivalently, the result is the *hidden polynomial state*

$$\rho_h := \sum_{y \in \mathbb{F}_q^d} \frac{|L_y^h|}{q^d} |L_y^h\rangle\langle L_y^h|. \tag{189}$$

Notice that these states are quite similar to the hidden subgroup states [Eq. (113)], modulo the fact that level sets of a polynomial can have different sizes unlike the cosets of a subgroup. Just as we upper bounded the query complexity of the HSP by analyzing the statistical distinguishability of the states [Eq. (113)], so we can upper bound the query complexity of the hidden polynomial problem by doing the same for the states [Eq. (189)]. Following a similar argument as in Sec. VII.E, one can show that

$$F(\rho_h, \rho_{h'})^2 \leqslant \frac{1}{q^d} \sum_{y,y' \in \mathbb{F}_q} \frac{|L_y^h \cap L_{y'}^h|^2}{|L_y^h|} \tag{190}$$

[cf. Eq. (131)]. Thus, the hidden polynomial states are pairwise distinguishable provided their level sets do not intersect too much. Since almost all polynomials are *absolutely irreducible* (i.e., they do not have any nontrivial factors even over an extension of the base field), this suffices to show that if the dimension d and the maximum degree of the polynomials are fixed, then the query complexity of the hidden polynomial problem is poly$(\log q)$ for almost all polynomials (Childs *et al.*, 2007).

Moving beyond query complexity, we would like to know whether there is an efficient quantum algorithm—i.e., one with running time poly$(\log q)$—for the hidden polynomial problem. Just as for the HSP, the most general version of this question is currently open. However, suppose we are promised that the hidden polynomial has the form

$$h(x_1, \ldots, x_d) = g(x_1, \ldots, x_{d-1}) - x_d \tag{191}$$

for some $(d-1)$-variate polynomial $g(x_1, \ldots, x_{d-1}) \in \mathbb{F}_q[x_1, \ldots, x_{d-1}]$. [A simple example is the *hidden parabola problem*, in which $h(x, y) = \alpha x^2 + \beta x - y$ for some unknown $\alpha, \beta \in \mathbb{F}_q$ that we would like to determine.] For such a hidden polynomial, the level sets are simply translates of each other, namely, $L_y^h = L_0^h + (0, \ldots, 0, y)$. Provided the maximum degree of the polynomial is at most some fixed constant, there is a quantum algorithm that determines h (up to an additive offset) in time poly$(d \log q)$ (Decker *et al.*, 2009). This algorithm is based on the pretty good measurement approach described in Sec. VII.G. Recall that the implementation of

the PGM relies on quantum sampling from the solutions of an average-case algebraic problem. For the hidden polynomial problem with a polynomial of the form in Eq. (191), this problem is a system of polynomial equations, much like the pair of quadratic equations [Eqs. (150) and (151)] that arises in the algorithm for the HSP in the Heisenberg group.

B. Shifted subset problems and exponential sums

Other families of hidden nonlinear structure problems arise in the setting of *shifted subset problems*. Such problems are most naturally stated directly in terms of quantum state distinguishability.[17] Suppose that for fixed subsets S, $T\subseteq\mathbb{F}_q^d$, we are given the quantum state $|S+t\rangle$ (a uniform superposition over the elements of $S+t$), where t is chosen uniformly at random from T. In other words, we are given the mixed quantum state

$$\rho_{S,T}=\frac{1}{|T|}\sum_{t\in T}|S+t\rangle\langle S+t|. \tag{192}$$

In the shifted subset problem, the goal is to determine some property of S or T or both using samples of $\rho_{S,T}$.

Childs *et al.* (2007) considered two examples of shifted subset problems in which the set S is a d-dimensional sphere, i.e., the set of points

$$\mathcal{S}_r := L_r^{\sum_{i=1}^d x_i^2}=\left\{x\in\mathbb{F}_q^d:\sum_{i=1}^d x_i^2=r\right\} \tag{193}$$

for some $r\in\mathbb{F}_q$. In the *hidden radius problem*, $T=\mathbb{F}_q^d$, and the goal is to learn r. In the *hidden flat of centers problem*, we are promised that $r=1$ and T is some unknown flat in $\mathbb{F}_q^d$; the goal is to determine this flat.

In general, when $T=\mathbb{F}_q^d$, symmetry ensures that $\rho_{S,T}$ is diagonal in the Fourier basis. Then the goal is to learn S from samples of a distribution given by its Fourier transform (recall Sec. III.D), namely,

$$\Pr(k)=\frac{1}{q^d|S|}\left|\sum_{x\in S}\omega_p^{\mathrm{Tr}(kx)}\right|^2, \tag{194}$$

where p is the characteristic of $\mathbb{F}_q$ and $\mathrm{Tr}:\mathbb{F}_q\to\mathbb{F}_p$ denotes the trace map. In particular, when $S=\mathcal{S}_r$ is a d-dimensional sphere, the distribution is proportional to an exponential sum known as a *Kloosterman sum* for d even or a *Salié sum* (a kind of twisted Kloosterman sum) for d odd. In either case, these distributions are information-theoretically distinguishable for different values of r. Moreover, a closed-form expression for Salié sums gives an efficient quantum algorithm for determining whether r is a quadratic residue, provided d is odd.

Suppose S is fixed and T is an unknown flat or, more generally, some low-degree surface. If we could perform the transformation $|S+t\rangle\mapsto|t\rangle$, then we could sample from points on the flat and thereby reconstruct it. Unfortunately, this transformation is generally not unitary since S could intersect with its translates. However, we can attempt to approximate such a transformation using the continuous-time quantum walk on the Cayley graph of $\mathbb{F}_q^d$ generated by S. When $S=\mathcal{S}_1$, this Cayley graph is known as the *Winnie Li graph*. Its eigenvalues are given by Kloosterman or Salié sums, depending on whether d is even or odd. For d odd, the explicit expression for Salié sums provides an efficient implementation of the quantum walk, which in turn gives an efficient quantum algorithm for the hidden flat of centers problem.

Of course, it is possible to make many other choices for S and T, so the above examples just begin to explore potential quantum algorithms for shifted subset problems. However, these simple examples already reveal a connection between the calculation of exponential sums[18] and the implementation of quantum walk that could perhaps be developed further. It would also be interesting to find concrete algorithmic applications of shifted subset problems.

C. Polynomial reconstruction by Legendre symbol evaluation

The quantum algorithm for the shifted Legendre symbol problem (Sec. VIII.D) recovers the constant term s of a linear function $f(x)=x+s$ hidden in the black-box function $\chi(f(x))=\chi(x+s)$, where χ is the Legendre symbol. As a precursor to the efficient quantum algorithm, it was shown that the quantum query complexity is $O(1)$, while the classical query complexity is $\Omega(\log p)$ (van Dam, 2002). Here we discuss the generalization to a nonlinear function $f(x)$ hidden in the black-box function $\chi(f(x))$. Russell and Shparlinski (2004) showed that the quantum query complexity is significantly lower than the classical query complexity even in this more general case. Whether there exists an efficient quantum algorithm to reconstruct the polynomial remains open.

Let $f\in\mathbb{F}_p[x]$ be an unknown polynomial. Given a black box for $\chi(f(x))$, with χ the Legendre symbol over $\mathbb{F}_p$, we want to reconstruct f using as few queries as possible. Note that for any $c\in\mathbb{F}_p^\times$, $\chi(c^2f(x))=\chi(f(x))$, making it impossible to tell the difference between $f(x)$ and $c^2f(x)$ on the basis of the black box $\chi(f(x))$. Moreover, if the factorization of $f(x)$ contains a square, i.e., if $f(x)=g^2(x)\cdot h(x)$, then $\chi(f(x))=\chi(g^2(x))\chi(h(x))$, which is identical to $\chi(h(x))$ (except possibly at the zeros of g). Thus we restrict our attention to polynomials that are monic and squarefree.

[17] Although the construction is technical, it is possible to formulate shifted subset problems in terms of a black box from which the state $\rho_{S,T}$ can be efficiently prepared on a quantum computer, but that typically must be queried exponentially many times to determine S, T on a classical computer (Childs *et al.*, 2007).

[18] Computing exponential sums is also closely related to counting the solutions of finite field equations. Kedlaya's algorithm (Sec. IV.H) can be used to efficiently approximate Kloosterman sums when the field characteristic is small [see Childs *et al.* (2007)].

In the case where $f(x)=x+s$, the reason that $O(1)$ quantum queries suffice is that the states $\Sigma_x\chi(x+s)|x\rangle$ are nearly orthogonal for different values of $s\in\mathbb{F}_p$. This follows from the identity

$$\sum_{x\in\mathbb{F}_p}\chi(x+r)\chi(x+s)=\begin{cases}p-1, & s=r\\ -1, & s\neq r.\end{cases} \tag{195}$$

For polynomials f,g of degree d that are monic and squarefree, the generalization of this fact is provided by the Weil bound (Lidl and Niederreiter, 1997), which implies that

$$\sum_{x\in\mathbb{F}_p}\chi(f(x))^2\geqslant p-d, \tag{196}$$

$$\sum_{x\in\mathbb{F}_p}\chi(f(x))\chi(g(x))\leqslant 2d\sqrt{p}\quad \text{if } f\neq g. \tag{197}$$

Note that for $d\geqslant\sqrt{p}/2$, Eq. (197) is trivial. However, for $d\leqslant p^{1/2-\varepsilon}$ with $\varepsilon>0$, we find the following.

Given a black-box function $\chi(f(x))$ where $f\in\mathbb{F}_p[x]$ is an unknown monic squarefree polynomial of degree d, two queries can be used to create the state

$$|\tilde{\chi}(f)\rangle:=\frac{1}{\sqrt{p}}\sum_{x\in\mathbb{F}_p}\tilde{\chi}(f(x))|x\rangle, \tag{198}$$

where $\tilde{\chi}$ is identical to χ except that $\tilde{\chi}(0)=1$. (This adjustment to the Legendre symbol is required to deal with the otherwise zero amplitudes for the zeros of f.) Using Eqs. (196) and (197), it follows that

$$|\langle\tilde{\chi}(f)|\tilde{\chi}(g)\rangle|\leqslant\frac{2d}{\sqrt{p}}. \tag{199}$$

Since there are p^d monic polynomials of degree d over $\mathbb{F}_p$, Theorem 3 [and specifically, Eq. (127)] shows that there is a measurement on $O(d)$ copies of $|\tilde{\chi}(f)\rangle$ that determines the d unknown coefficients of f with probability $1-O(1/p)$. The classical query complexity of this problem can be shown to be $\Omega(d\log p)$, which therefore gives a separation between classical and quantum query complexity.

X. APPROXIMATING #P-COMPLETE PROBLEMS

Recently, there has been considerable interest in quantum algorithms for approximately solving various #P-complete problems. The first such algorithms were for approximating the Jones polynomial; more recently, similar ideas have been used to give approximate solutions to other #P-complete problems. These algorithms are not as closely related to Shor's as most of those discussed here, but they are decidedly algebraic, relying heavily on group representation theory.

The *Jones polynomial* is a central object in low-dimensional topology with surprising connections to physics. Witten (1989) showed that the Jones polynomial is closely related to topological quantum field theory (TQFT). Freedman *et al.* (2003) investigated the relationship between TQFT and topological quantum computing, showing that quantum computers can efficiently simulate TQFTs (Freedman, Kitaev, and Wang, 2002) and that in fact TQFTs essentially capture the power of quantum computation (Freedman, Larsen, and Wang, 2002). In particular, Freedman, Kitaev, and Wang (2002) showed that quantum computers can efficiently approximate the Jones polynomial at a fifth root of unity. Subsequently, Aharonov *et al.* (2009) described an explicit quantum algorithm for approximating the Jones polynomial, generalizing to any primitive root of unity [see also Wocjan and Yard (2008)].

To define the Jones polynomial, we must first introduce the concepts of knots and links. A *knot* is an embedding of the circle in $\mathbb{R}^3$, i.e., a closed loop of string that may wrap around itself in any way. More generally, a *link* is a collection of any number of knots that may be intertwined. In an oriented link, each loop of string is directed. It is natural to identify links that are isotopic, i.e., that can be transformed into one another by continuous deformation of the strings.

The Jones polynomial of an oriented link L is a Laurent polynomial $V_L(t)$ in the variable $\sqrt{t}$, i.e., a polynomial in $\sqrt{t}$ and $1/\sqrt{t}$. The polynomial is a *link invariant*, meaning that $V_L(t)=V_{L'}(t)$ if the oriented links L and L' are isotopic. While it is possible for the Jones polynomial to take the same value on two nonisotopic links, it can often distinguish links; for example, the Jones polynomials of the two orientations of the trefoil knot are different.

Given an oriented link L, one way to define its Jones polynomial is as follows (Kauffman, 1987). We define the Kauffman bracket $\langle L\rangle$, which does not depend on the orientation of L. Each crossing in the link diagram can be opened in one of two ways, and for any given crossing we have

(200)

where the rest of the link remains unchanged. Repeatedly applying this rule, we eventually arrive at a link consisting of disjoint unknots. The Kauffman bracket of a single unknot is $\langle\bigcirc\rangle:=1$ and, more generally, the Kauffman bracket of n unknots is $(-t^{1/2}-t^{-1/2})^{n-1}$. By itself, the Kauffman bracket is not a link invariant, but it can be turned into one by taking into account the orientation of the link, giving the Jones polynomial. For any oriented link L, we define its *writhe* $w(L)$ as the number of crossings of the form  minus the number of crossings of the form  . Then the Jones polynomial is defined as

$$V_L(t):=(-t^{-1/4})^{3w(L)}\langle L\rangle. \tag{201}$$

It is useful to view links as arising from *braids*. A braid is a collection of n parallel strands, with adjacent strands allowed to cross over or under one another. Two braids on the same number of strands can be composed by placing them end to end. The *braid group* B_n on n strands is an infinite group with generators $\{\sigma_1,\ldots,\sigma_{n-1}\}$, where σ_i denotes a twist in which strand i passes over

strand $i+1$, interchanging the two strands. More formally, the braid group is defined by the relations $\sigma_i\sigma_{i+1}\sigma_i=\sigma_{i+1}\sigma_i\sigma_{i+1}$ and $\sigma_i\sigma_j=\sigma_j\sigma_i$ for $|i-j|>1$.

Braids and links differ in that the ends of a braid are open, whereas a link consists of closed strands. We can obtain a link from a braid by connecting the ends of the strands in some way. One simple way to close a braid is via the *trace closure*, in which the ith strand of one end is connected to the ith strand of the other end for each $i=1,\ldots,n$, without crossing the strands. A theorem of Alexander (1923) states that any link can be obtained as the trace closure of some braid.

The Jones polynomial of the trace closure of a braid can be expressed in terms of the Markov trace (a weighted variant of the usual trace) of a representation of the braid group defined over the Temperley-Lieb algebra (Jones, 1985). When evaluating the Jones polynomial $V_L(t)$ at the root of unity $t=e^{2\pi i/k}$, this representation is unitary. This naturally suggests a quantum algorithm for approximating the Jones polynomial. Suppose that we can implement unitary operations corresponding to twists of adjacent strands on a quantum computer. By composing such operations, we can implement a unitary operation corresponding to the entire braid. It remains to approximate the Markov trace of this operator.

The trace of a unitary operation U can be approximated on a quantum computer using the *Hadamard test*. If a conditional U operation is applied to the state $|+\rangle\otimes|\psi\rangle$ and the first qubit is measured in the $|\pm\rangle$ basis, where $|\pm\rangle:=(|0\rangle\pm|1\rangle)/\sqrt{2}$, the expectation value of the outcome is precisely $\mathrm{Re}(\langle\psi|U|\psi\rangle)$. (This is simply the phase estimation procedure described in Sec. III.C with $n=1$, i.e., with a single bit of precision.) Replacing the states $|\pm\rangle$ by the states $|\pm i\rangle:=(|0\rangle\pm i|1\rangle)/\sqrt{2}$, we can approximate $\mathrm{Im}(\langle\psi|U|\psi\rangle)$. Using a maximally mixed state as input instead of the pure state $|\psi\rangle$ and sampling sufficiently many times from the resulting distribution, we can obtain an approximation of $\mathrm{Re}(\mathrm{Tr}\,U)$ or $\mathrm{Im}(\mathrm{Tr}\,U)$. Similarly, we can approximate a weighted trace by sampling from an appropriate distribution over pure states.

Applying this approach to the relevant unitary representation of the braid group, one obtains a quantum algorithm for approximating the Jones polynomial of the trace closure of a braid at a root of unity. In particular, for a braid on n strands, with m crossings and $t=e^{2\pi i/k}$, there is an algorithm running in time $\mathrm{poly}(n,m,k)$ that outputs an approximation differing from the actual value $V_L(t)$ of the Jones polynomial by at most $(2\cos\pi/k)^{n-1}/\mathrm{poly}(n,k,m)$, with only exponentially small probability of failure (Aharonov *et al.*, 2009).

Given a braid with an even number of strands, another natural way to create a link is called the *plat closure*. Here we simply join adjacent pairs of strands at each end of the braid. The plat closure can be viewed as the trace closure of a braid on $2n$ strands together with $2n$ additional straight strands. Using this fact, we can express the Jones polynomial of the plat closure of a braid at $t=e^{2\pi i/k}$ as the expectation value of a particular unitary representation of the braid group in a pure quantum state. Thus the Jones polynomial of the plat closure can also be approximated using the Hadamard test, but now using a pure input state instead of a mixed one. This gives an efficient quantum algorithm for an additive approximation of the Jones polynomial of the plat closure of a braid at a root of unity (Aharonov *et al.*, 2009).

Note that these algorithms only provide *additive* approximations, meaning that the error incurred by the algorithm is independent of the value being approximated, which is undesirable when that value is small. (In fact, note that the additive error increases exponentially with n, the number of strands in the braid.) It would be preferable to obtain a *multiplicative* approximation or, better still, an exact calculation. However, exactly computing the Jones polynomial is #P-hard (Jaeger *et al.*, 1990) and hence unlikely to be possible even with a quantum computer. Furthermore, obtaining the additive approximation achieved by Aharonov *et al.* (2009) for the Jones polynomial of the plat closure of a braid is as hard as any quantum computation (Freedman, Larsen, and Wang, 2002; Bordewich *et al.*, 2005; Aharonov and Arad, 2006; Wocjan and Yard, 2008).

To implement the quantum algorithm for approximating the trace closure of a braid, it is only necessary to have a single pure qubit (the qubit initialized to $|+\rangle$ in the Hadamard test) and many mixed ones. Thus it can be carried out in the one clean qubit model introduced by Knill and Laflamme (1998) to investigate the power of mixed state quantum computation. In fact, the problem of estimating the Jones polynomial of the trace closure of a braid at a fifth root of unity (to the precision described above) exactly characterizes the power of this model (Jordan and Wocjan, 2008; Shor and Jordan, 2008), just as the approximation of the plat closure characterizes general quantum computation.

We conclude by mentioning various extensions of these results. Wocjan and Yard (2008) showed how to evaluate the Jones polynomial of a generalized closure of a braid and how to evaluate a generalization of the Jones polynomial called the HOMFLYPT polynomial. Recent work of Aharonov, Arad, *et al.* (2007) showed how to approximate the Tutte polynomial of a planar graph, which in particular gives an approximation of the partition function of the Potts model on a planar graph; this problem also characterizes the power of quantum computation, albeit only for unphysical choices of parameters. More generally, there are efficient quantum algorithms to compute additive approximations of tensor networks (Arad and Landau, 2008).

ACKNOWLEDGMENTS

We thank Sean Hallgren for discussions of algorithms for number fields. We also thank Dorit Aharonov, Greg Kuperberg, Frédéric Magniez, Cris Moore, Miklos Santha, John Watrous, and Pawel Wocjan for comments on a preliminary version. This article was written in part while A.M.C. was at the Institute for Quantum Informa-

tion at Caltech, where he received support from the National Science Foundation under Grant No. PHY-456720 and from the U.S. Army Research Office under Grant No. W9111NF-05-1-0294. A.M.C. was also supported in part by MITACS, NSERC, and the U.S. ARO/DTO. W.v.D. was supported in part by the NSA under U.S. Army Research Office Contract Nos. W911NF-04-R-0009 and W911NF-04-R-0001 and by an NSF CAREER award.

APPENDIX A: NUMBER THEORY

1. Arithmetic mod N

When performing calculations with integers mod N we use the equivalence relation $x=y \bmod N$ if and only if $x-y \in N\mathbb{Z}=\{\ldots,-N,0,N,2N,\ldots\}$. Often we omit the notation mod N and instead consider x and y as elements of the ring $\mathbb{Z}/N\mathbb{Z}$. Other ways of denoting this ring are $\mathbb{Z}_N$ and $\mathbb{Z}/(N)$; throughout we use the notation $\mathbb{Z}/N\mathbb{Z}$, which is conventional in computational number theory. Although formally the elements of $\mathbb{Z}/N\mathbb{Z}$ are the sets $\{\ldots,-N+x,x,x+N,x+2N,\ldots\}$, we often simply represent such an element by the integer x; this representation is unique if we require $x \in \{0,\ldots,N-1\}$.

Addition mod N corresponds to the additive group $(\mathbb{Z}/N\mathbb{Z},+)$, which has N elements. For example, with $N=2$ we have $0+0=0$, $1+0=0+1=1$, and $1+1=0$. If N has the prime factorization $N=p_1^{r_1}\cdots p_k^{r_k}$, then the additive group $\mathbb{Z}/N\mathbb{Z}$ can be decomposed as $(\mathbb{Z}/p_1^{r_1}\mathbb{Z})\times\cdots\times(\mathbb{Z}/p_k^{r_k}\mathbb{Z})$.

Multiplication mod N is more complicated than addition as not all elements of $\mathbb{Z}/N\mathbb{Z}$ have a multiplicative inverse. For example, $5\times 5=1 \bmod 6$, but there is no element x such that $2x=1 \bmod 6$. In general, there exists a y such that $xy=1 \bmod N$ if and only if $\gcd(x,N)=1$, where $\gcd(x,N)$ is the *greatest common divisor* of x and N. The set of such invertible elements of $\mathbb{Z}/N\mathbb{Z}$ makes up the multiplicative group $(\mathbb{Z}/N\mathbb{Z})^\times$. It is easy to check that in $\mathbb{Z}/6\mathbb{Z}$ there are only two invertible elements: $(\mathbb{Z}/6\mathbb{Z})^\times=\{1,5\}$. The size of the multiplicative group $(\mathbb{Z}/N\mathbb{Z})^\times$ depends on the prime factorization of N; one can show that for $N=p_1^{r_1}\cdots p_k^{r_k}$,

$$\varphi(N) := |(\mathbb{Z}/N\mathbb{Z})^\times| = (p_1-1)p_1^{r_1-1}\cdots(p_k-1)p_k^{r_k-1}, \tag{A1}$$

where φ is called *Euler's totient function*. Similarly to the additive case, one also has the multiplicative group isomorphism $(\mathbb{Z}/N\mathbb{Z})^\times \cong (\mathbb{Z}/p_1^{r_1}\mathbb{Z})^\times\times\cdots\times(\mathbb{Z}/p_k^{r_k}\mathbb{Z})^\times$.

By combining the isomorphisms for the additive and multiplicative groups of integers mod N, we obtain the *Chinese remainder theorem*. This states that for $N=p_1^{r_1}\cdots p_k^{r_k}$, the bijection between the elements of $x \in \mathbb{Z}/N\mathbb{Z}$ and the k-tuples $(x_1,\ldots,x_k) \in (\mathbb{Z}/p_1^{r_1}\mathbb{Z})\times\cdots\times(\mathbb{Z}/p_k^{r_k}\mathbb{Z})$ (with $x_i=x \bmod p_i^{k_i}$ for all i) respects both addition and multiplication in the ring $\mathbb{Z}/N\mathbb{Z}$. This fact often allows us to break up algebraic problems in $\mathbb{Z}/N\mathbb{Z}$ into k smaller problems in $\mathbb{Z}/p_i^{r_i}\mathbb{Z}$, which can be easier to deal with.

2. Finite fields and their extensions

For a prime number p we have $\varphi(p)=p-1$, which means that all but the zero element of $\mathbb{Z}/p\mathbb{Z}$ have a multiplicative inverse mod p. Thus $\mathbb{Z}/p\mathbb{Z}$ is a *finite field*, which we denote by $\mathbb{F}_p$. Just as $\mathbb{R}$ is a field that can be extended to $\mathbb{C}$ by including the solutions to polynomial equations such as $\alpha^2+1=0$, so can the finite field $\mathbb{F}_p$ be extended to $\mathbb{F}_{p^r}$ for any positive integer r. Any finite field has order $q=p^r$ with p some prime, and for each prime power p^r there is a finite field of that order. Up to isomorphism, this finite field is in fact unique, so we can refer to *the* finite field $\mathbb{F}_q$ without ambiguity. The additive group of $\mathbb{F}_{p^r}$ is isomorphic to the additive group $(\mathbb{Z}/p\mathbb{Z})^r$, while the multiplicative group $\mathbb{F}_{p^r}^\times$ is cyclic and is isomorphic to the additive group $\mathbb{Z}/(p^r-1)\mathbb{Z}$. Note that $\mathbb{F}_{p^r}$ is very different from $\mathbb{Z}/p^r\mathbb{Z}$ for $r>1$, as $|\mathbb{F}_{p^r}^\times|=p^r-1$ while $|(\mathbb{Z}/p^r\mathbb{Z})^\times|=(p-1)p^{r-1}$.

A standard way of explicitly constructing a finite field $\mathbb{F}_{p^r}$ is by extending $\mathbb{F}_p$ with a formal variable α satisfying $T(\alpha)=0$, where T is an irreducible polynomial of degree r in $\mathbb{F}_p[\alpha]$. The finite field $\mathbb{F}_{p^r}$ is isomorphic to the ring of polynomials $\mathbb{F}_p[\alpha]$ modulo the polynomial $T(\alpha)$, i.e., $\mathbb{F}_{p^r}\cong\mathbb{F}_p[\alpha]/T(\alpha)$.

Example (*Construction of* $\mathbb{F}_8$). Mod 2, the polynomial $T(\alpha)=\alpha^3+\alpha+1$ is irreducible: $T(\alpha)$ cannot be written as the product of two nontrivial polynomials. Hence $\mathbb{F}_2[\alpha]/(\alpha^3+\alpha+1)$ is the finite field $\mathbb{F}_8$. The addition in this field is the straightforward addition of quadratic polynomials mod 2, such that, for example, $(\alpha^2+\alpha)+(\alpha^2+1)=\alpha+1$. Multiplication of the elements is slightly more involved, but the explicit multiplication table of Table II confirms that $\mathbb{F}_2[\alpha]/(\alpha^3+\alpha+1)$ is indeed a field. Note, for example, that α has multiplicative inverse α^2+1 as $\alpha(\alpha^2+1)=\alpha^3+\alpha=1$ by the equality $\alpha^3+\alpha+1=0$.

Obviously $\mathbb{F}_8$ contains the subfield $\mathbb{F}_2$, but less obviously $\mathbb{F}_8$ does not contain $\mathbb{F}_4$. In general, $\mathbb{F}_{q_1}$ contains the finite field $\mathbb{F}_{q_2}$ if and only if q_1 is a power of q_2, hence if and only if $q_1=p^{r_1}$ and $q_2=p^{r_2}$ where r_2 divides r_1. For a finite field $\mathbb{F}_q$ with $q=p^r$ and p prime, we call $\mathbb{F}_p$ the *base field* of $\mathbb{F}_q$, and we call $\mathbb{F}_{p^r}$ the *degree r extension* of the field $\mathbb{F}_p$. By taking the limit of arbitrarily high degree r, we obtain the algebraic closure $\bar{\mathbb{F}}_p$ of $\mathbb{F}_p$, which is an infinite field.

Although the construction of an extension field using an irreducible polynomial makes it easy to explicitly perform calculations, the procedure soon becomes cumbersome, as Table II already shows. Furthermore, the representation depends on the specific polynomial being used, so it introduces a certain arbitrariness. Hence, whenever possible we talk about finite fields without specifying a particular representation.

TABLE II. The multiplication table of the finite field $\mathbb{F}_8$ represented by the elements of $\mathbb{F}_2[\alpha]/(\alpha^3+\alpha+1)$.

$\times$	0	1	α	$\alpha+1$	α^2	α^2+1	$\alpha^2+\alpha$	$\alpha^2+\alpha+1$
0	0	0	0	0	0	0	0	0
1	0	1	α	$\alpha+1$	α^2	α^2+1	$\alpha^2+\alpha$	$\alpha^2+\alpha+1$
α	0	α	α^2	$\alpha^2+\alpha$	$\alpha+1$	1	$\alpha^2+\alpha+1$	α^2+1
$\alpha+1$	0	$\alpha+1$	$\alpha^2+\alpha$	α^2+1	$\alpha^2+\alpha+1$	α^2	1	α
α^2	0	α^2	$\alpha+1$	$\alpha^2+\alpha+1$	$\alpha^2+\alpha$	α	α^2+1	1
α^2+1	0	α^2+1	1	α^2	α	$\alpha^2+\alpha+1$	$\alpha+1$	$\alpha^2+\alpha$
$\alpha^2+\alpha$	0	$\alpha^2+\alpha$	$\alpha^2+\alpha+1$	1	α^2+1	$\alpha+1$	α	α^2
$\alpha^2+\alpha+1$	0	$\alpha^2+\alpha+1$	α^2+1	α	1	$\alpha^2+\alpha$	α^2	$\alpha+1$

3. Structure of finite fields

Starting from the infinite field $\bar{\mathbb{F}}_p$, the elements of $\mathbb{F}_{p^r}$ can be characterized as the $q=p^r$ solutions to the equation $x^q=x$. This immediately implies the above statement that $\mathbb{F}_{p^{r_1}}$ contains $\mathbb{F}_{p^{r_2}}$ if and only r_2 divides r_1.

Within the finite field $\mathbb{F}_{p^r}$, the *Frobenius automorphism* $\phi:\mathbb{F}_{p^r}\to\mathbb{F}_{p^r}$ is the map defined by $\phi(x)=x^p$. It is a field automorphism, meaning that $\phi(x+y)=\phi(x)+\phi(y)$ and $\phi(xy)=\phi(x)\phi(y)$ for all $x,y\in\mathbb{F}_{p^r}$. Iterating the Frobenius automorphism gives the r different maps $\phi^j:x\mapsto x^{p^j}$ for $j=0,1,\ldots,r-1$, which are all automorphisms of $\mathbb{F}_{p^r}$. Because $\phi(a)=a$ for all base field elements $a\in\mathbb{F}_p$, we see that if $x\in\mathbb{F}_{p^r}$ is a root of a polynomial $F(X)=a_dX^d+\cdots+a_1X+a_0\in\mathbb{F}_p[X]$ with coefficients in the base field $\mathbb{F}_p$, then so are its conjugates $\phi^j(x)$ as, assuming $F(x)=0$, we have $F(\phi(x))=\Sigma_i a_i(\phi(x))^i=\Sigma_i\phi(a_ix^i)=\phi(F(x))=0$. This result generalizes to multivariate polynomials $F\in\mathbb{F}_p[X_1,\ldots,X_n]$ with roots $x=(x_1,\ldots,x_n)\in\mathbb{F}_{p^r}^n$: if $F(x)=0$ then also $F(\phi^j(x))=F(\phi^j(x_1),\ldots,\phi^j(x_n))=0$. Hence the set of solutions $\{x\in\mathbb{F}_{p^r}^n:F(x)=0\}$ is invariant under the Frobenius automorphism.

APPENDIX B: REPRESENTATION THEORY OF FINITE GROUPS

In this appendix, we review the theory of group representations needed to study the non-Abelian HSP. Here we restrict our attention to finite groups and to representations over finite-dimensional complex vector spaces. For a more detailed introduction to representation theory, see Hamermesh (1989) and Serre (1977).

1. General theory

A *linear representation* (or simply *representation*) of a finite group G over the vector space $\mathbb{C}^n$ is a homomorphism $\sigma:G\to\mathsf{GL}(\mathbb{C}^n)$, i.e., a map from group elements to nonsingular $n\times n$ complex matrices satisfying $\sigma(x)\sigma(y)=\sigma(xy)$ for all $x,y\in G$. Clearly, $\sigma(1)=1$ and $\sigma(x^{-1})=\sigma(x)^{-1}$. We say that $\mathbb{C}^n$ is the *representation space* of σ, where n is called its dimension (or degree) and is denoted d_σ.

Two representations σ and σ' with representation spaces $\mathbb{C}^n$ are *isomorphic* (denoted $\sigma\sim\sigma'$) if and only if there is an invertible linear transformation $M\in\mathbb{C}^{n\times n}$ such that $M\sigma(x)=\sigma'(x)M$ for all $x\in G$. (Representations of different dimensions cannot be isomorphic.) Every representation is isomorphic to a unitary representation, i.e., one for which $\sigma(x)^{-1}=\sigma(x)^\dagger$ for all $x\in G$. Thus we can restrict our attention to unitary representations without loss of generality.

The simplest representations are those of dimension 1, such that $\sigma(x)\in\mathbb{C}$ with $|\sigma(x)|=1$ for all $x\in G$. Every group has a one-dimensional representation called the *trivial representation*, defined by $\sigma(x)=1$ for all $x\in G$.

Two particularly useful representations of a group G are its *left regular representation* and its *right regular representation*. Both of these representations have dimension $|G|$, and their representation space is the group algebra $\mathbb{C}G$, i.e., the $|G|$-dimensional complex vector space spanned by basis vectors $|x\rangle$ for $x\in G$. The left regular representation L satisfies $L(x)|y\rangle=|xy\rangle$, and the right regular representation R satisfies $R(x)|y\rangle=|yx^{-1}\rangle$. In particular, both regular representations are *permutation representations* as each consists entirely of permutation matrices.

Given two representations $\sigma:G\to V$ and $\sigma':G\to V'$, we can define their direct sum, a representation $\sigma\oplus\sigma':G\to V\oplus V'$ of dimension $d_{\sigma\oplus\sigma'}=d_\sigma+d_{\sigma'}$. The representation matrices of $\sigma\oplus\sigma'$ are of the form

$$(\sigma\oplus\sigma')(x)=\begin{pmatrix}\sigma(x) & 0\\ 0 & \sigma'(x)\end{pmatrix} \tag{B1}$$

for all $x\in G$.

A representation is *irreducible* if it cannot be decomposed as the direct sum of two other representations. Any representation of a finite group G can be written as a direct sum of irreducible representations (or *irreps*) of G. Up to isomorphism, G has a finite number of irreps. The symbol $\hat{G}$ denotes a complete set of irreps of G, one for each isomorphism type.

Another way to combine two representations is with the tensor product. The tensor product of $\sigma:G\to V$ and $\sigma':G\to V'$ is $\sigma\otimes\sigma':G\to V\otimes V'$, a representation of dimension $d_{\sigma\otimes\sigma'}=d_\sigma d_{\sigma'}$.

The *character* of a representation σ is the function $\chi_\sigma:G\to\mathbb{C}$ defined by $\chi_\sigma(x):=\mathrm{Tr}\,\sigma(x)$. We have $\chi_\sigma(1)$

$=d_\sigma$, $\chi(x^{-1})=\chi(x)^*$, and $\chi(yx)=\chi(xy)$ for all $x,y\in G$. For two representations σ,σ', we have $\chi_{\sigma\oplus\sigma'}=\chi_\sigma+\chi_{\sigma'}$ and $\chi_{\sigma\otimes\sigma'}=\chi_\sigma\cdot\chi_{\sigma'}$.

Perhaps the most useful result in representation theory is Schur's lemma, which can be stated as follows.

Theorem 4 (Schur's lemma). Let σ and σ' be two irreducible representations of G and let $M\in\mathbb{C}^{d_\sigma\times d_{\sigma'}}$ be a matrix satisfying $\sigma(x)M=M\sigma'(x)$ for all $x\in G$. Then if $\sigma\not\sim\sigma'$ we have $M=0$ and if $\sigma=\sigma'$, then M is a scalar multiple of the identity matrix.

Schur's lemma can be used to prove the following orthogonality relation for irreducible representations.

Theorem 5. For two irreps $\sigma,\sigma'\in\hat{G}$, we have

$$\frac{d_\sigma}{|G|}\sum_{x\in G}\sigma(x)^*_{i,j}\sigma'(x)_{i',j'}=\delta_{\sigma,\sigma'}\delta_{i,i'}\delta_{j,j'}, \tag{B2}$$

where $\delta_{\sigma,\sigma'}$ *is* 1 *if* $\sigma=\sigma'$ and 0 otherwise. In particular, this implies a corresponding orthogonality relation for the irreducible characters (i.e., the characters of the irreducible representations):

Theorem 6. For two irreps $\sigma,\sigma'\in\hat{G}$, we have

$$(\chi_\sigma,\chi_{\sigma'}):=\frac{1}{|G|}\sum_{x\in G}\chi_\sigma(x)^*\chi_{\sigma'}(x)=\delta_{\sigma,\sigma'}. \tag{B3}$$

Characters provide a simple test for irreducibility. In particular, for any representation σ, $(\chi_\sigma,\chi_\sigma)$ is a positive integer and equal to 1 if and only if σ is irreducible.

Any representation of G can be broken up into its irreducible components. The regular representations of G are useful for understanding such decompositions since they contain every possible irrep of G, each occurring a number of times equal to its dimension. In particular,

$$L\cong\bigoplus_{\sigma\in G}(\sigma\otimes 1_{d_\sigma}),\quad R\cong\bigoplus_{\sigma\in G}(1_{d_\sigma}\otimes\sigma^*), \tag{B4}$$

where 1_d denotes the $d\times d$ identity matrix. In fact, this holds with the same isomorphism for both L and R since they are commutants of each other. The isomorphism is simply the Fourier transform over G. For its precise definition, as well as a proof that it decomposes the regular representations, see Sec. VI.

Considering $\chi_L(1)=\chi_R(1)=|G|$ and using this decomposition, we find the well known identity

$$\sum_{\sigma\in\hat{G}}d_\sigma^2=|G|. \tag{B5}$$

Noting also that $\chi_L(x)=\chi_R(x)=0$ for any $x\in G\backslash\{1\}$, we see that

$$\sum_{\sigma\in\hat{G}}d_\sigma\chi_\sigma(x)=0. \tag{B6}$$

In general, the multiplicity of the irrep $\sigma\in\hat{G}$ in an arbitrary representation τ of G is given by $\mu_\sigma^\tau:=(\chi_\sigma,\chi_\tau)$. Then we have the decomposition

$$\tau\cong\bigoplus_{\sigma\in\hat{G}}(\sigma\otimes 1_{\mu_\sigma^\tau}). \tag{B7}$$

The projection onto the *σ-isotypic subspace* of τ is given by

$$\Pi_\sigma^\tau:=\frac{d_\sigma}{|G|}\sum_{x\in G}\chi_\sigma(x)^*\tau(x). \tag{B8}$$

Any representation σ of G can also be viewed as a representation of any subgroup $H\leqslant G$, simply by restricting its domain to elements of H. We denote the resulting *restricted representation* by $\mathrm{Res}_H^G\sigma$. Even when σ is irreducible over G, it will in general not be irreducible over H. (It is also possible to extend any representation σ' of H to an *induced representation* $\mathrm{Ind}_H^G\sigma'$ of G, but we will not need the definition here.)

We conclude with some examples of groups and their irreducible representations.

2. Abelian groups

The irreducible representations of any finite Abelian group are all one dimensional. (Conversely, any non-Abelian group has some irrep of dimension greater than 1.)

For a cyclic group $G=\mathbb{Z}/n\mathbb{Z}$, all irreps are of the form $\sigma_k:\mathbb{Z}/n\mathbb{Z}\to\mathbb{C}$ with $\sigma_k(x):=e^{2\pi ikx/n}$, where $k\in\mathbb{Z}/n\mathbb{Z}$ uniquely labels the representation. Hence there are indeed n inequivalent irreps of $\mathbb{Z}/n\mathbb{Z}$, all of dimension 1.

Any finite Abelian group can be written as a direct product of cyclic factors, and its irreducible representations are given by products of irreps of those factors. For example, the irreducible representations of the group $G=(\mathbb{Z}/n\mathbb{Z})^2$ are given by $\sigma_k(x):=e^{2\pi i(k_1x_1+k_2x_2)/n}$, where $k=(k_1,k_2)\in(\mathbb{Z}/n\mathbb{Z})^2$ uniquely labels the irrep.

3. Dihedral group

The dihedral group of order $2n$ is $D_n=\mathbb{Z}/n\mathbb{Z}\rtimes\mathbb{Z}/2\mathbb{Z}$, with the group law

$$(x,a)\cdot(y,b)=(x+(-1)^a y,a+b) \tag{B9}$$

for $x,y\in\mathbb{Z}/n\mathbb{Z}$ and $a,b\in\mathbb{Z}/2\mathbb{Z}$.

For n even, we have the following one-dimensional representations:

$$\sigma_{\mathrm{tt}}((x,a)):=1, \tag{B10}$$

$$\sigma_{\mathrm{ts}}((x,a)):=(-1)^a, \tag{B11}$$

$$\sigma_{\mathrm{st}}((x,a)):=(-1)^x, \tag{B12}$$

$$\sigma_{\mathrm{ss}}((x,a)):=(-1)^{x+a}; \tag{B13}$$

for n odd, we have only σ_{tt} and σ_{ts}. The two-dimensional representations are of the form

$$\sigma_h((x,0)) := \begin{pmatrix} e^{2\pi i h x/n} & 0 \\ 0 & e^{-2\pi i h x/n} \end{pmatrix} \tag{B14}$$

and

$$\sigma_h((x,1)) := \begin{pmatrix} 0 & e^{-2\pi i h x/n} \\ e^{-2\pi i h x/n} & 0 \end{pmatrix} \tag{B15}$$

for some $h \in \{1,2,\ldots,\lceil \frac{n}{2} \rceil - 1\}$. It is straightforward to check that these representations are all irreducible and that the sum of the dimensions squared gives $2n$.

APPENDIX C: CURVES OVER FINITE FIELDS

Kedlaya's quantum algorithm for counting the number of points on a curve over a finite field relies on several results in algebraic geometry. Here we explain some of the central concepts that are necessary to understand the algorithm. For concreteness, we limit ourselves to the case of planar algebraic curves. Our notation follows Lorenzini (1996); see this work for more information on this topic.

Given a bivariate polynomial $f \in \mathbb{F}_q[X,Y]$, we can consider the solutions to the equation $f(x,y)=0$ with x,y elements of the base field $\mathbb{F}_q$ or of an extension field $\mathbb{F}_{q^r}$. The set of these solutions is the planar curve denoted by $C_f(\mathbb{F}_q)$ or $C_f(\mathbb{F}_{q^r})$, respectively. Often we drop the subscript f when it is clear from context.

1. Affine and projective spaces

The theory of algebraic equations works more generally if we allow points at infinity to be possible solutions as well. We frequently work over the projective plane $\mathbb{P}^2$, which for a given finite field $\mathbb{F}_{q^r}$ can be expressed as

$$\mathbb{P}^2(\mathbb{F}_{q^r}) = (\mathbb{F}_{q^r}^3 \setminus \{(0,0,0)\})/\sim, \tag{C1}$$

where two points are equivalent, denoted $(x,y,z) \sim (x',y',z')$, if and only if there exists a $\lambda \in \mathbb{F}_{q^r}^\times$ such that $(\lambda x, \lambda y, \lambda z) = (x',y',z')$. These rays in $\mathbb{F}_q^3$ are denoted by $(x:y:z)$, i.e.,

$$(x:y:z) = \{(\lambda x, \lambda y, \lambda z) : \lambda \in \mathbb{F}_{q^r}^\times\} \subset \mathbb{F}_{q^r}^3 \tag{C2}$$

for all $(x,y,z) \neq (0,0,0)$.

One can easily verify that the projective plane $\mathbb{P}^2(\mathbb{F}_q)$ consists of q^2+q+1 points, of which q^2 lie in the affine plane $\mathbb{A}^2(\mathbb{F}_q) = \{(x,y,1) : (x,y) \in \mathbb{F}_q\}$; the remaining $q+1$ points are the line at infinity $\{(x,1,0) : x \in \mathbb{F}_q\}$ and the point at infinity $\{(1,0,0)\}$. This decomposition can be summarized by $\mathbb{P}^2 = \mathbb{A}^2 \cup \mathbb{P}^1 = \mathbb{A}^2 \cup \mathbb{A}^1 \cup \mathbb{A}^0$. For clarity, an affine space is often indicated by $\mathbb{A}^n(\mathbb{F}_q)$ rather than by the equivalent set $\mathbb{F}_q^n$, as the latter suggests a vector space with an origin, a concept that plays no role in affine spaces. In this article we ignore this subtlety.

2. Projective curves

The affine solutions to the polynomial equation $f(X,Y)=0$ over $\mathbb{F}_q$ consist of the set $\{(x,y) \in \mathbb{F}_q^2 : f(x,y) = 0\}$, but for the solutions in the projective plane $\mathbb{P}^2(\mathbb{F}_q)$ we must make the following adjustment. To define $f(X,Y)$ in $\mathbb{P}^2$, we introduce a third variable Z that allows us to translate f into a homogeneous polynomial, such that if $f(x,y,z)=0$ for $(x,y,z) \in \mathbb{F}_q^3 \setminus \{(0,0,0)\}$, then $f(\lambda x, \lambda y, \lambda z) = 0$ for all $\lambda \in \mathbb{F}_q^\times$. For example, with $f(X,Y) = Y^2 + X^3 + X + 1$, we have $f(X,Y,Z) = Y^2 Z + X^3 + XZ^2 + Z^3$.

In other words, an algebraic curve C_f in the projective plane is defined by a homogeneous polynomial $f \in \mathbb{F}_q[X,Y,Z]$ and its set of $\mathbb{F}_{q^r}$-rational solutions is given by

$$C_f(\mathbb{F}_{q^r}) = \{(x:y:z) : f(x,y,z) = 0\} \subset \mathbb{P}^2(\mathbb{F}_{q^r}). \tag{C3}$$

Note that for each extension degree r there is a different set of solutions $C_f(\mathbb{F}_{q^r})$. Explicit examples of curves are given in Secs. IV.F and IV.H.

3. Properties of curves

Let $f \in \mathbb{F}_q[X,Y,Z]$ define a planar, projective curve C_f. A point $(x:y:z) \in C_f(\mathbb{F}_{q^r})$ is called *nonsingular* if and only if

$$\left(\frac{\partial f}{\partial X}, \frac{\partial f}{\partial Y}, \frac{\partial f}{\partial Z}\right)(x,y,z) \neq (0,0,0), \tag{C4}$$

where $\partial f / \partial X$ denotes the formal derivative of f with respect to X. A projective curve is called *smooth* if all its points are nonsingular. In many ways, curves over finite fields are analogous to compact Riemann surfaces. Most importantly, one can assign a *genus* g to a smooth projective curve C_f, just as one can for a compact Riemann surface. (This is why we use the projective curve: the affine curve is not compact.) The projective line $\mathbb{P}^1$, defined by a linear equation such as $X=0$, has genus 0; elliptic curves, defined by cubic equations, have genus 1 and, in general, a degree d polynomial gives a curve with genus $g = \frac{1}{2}(d-1)(d-2)$. The complexity of algorithms for curves often depends critically on the genus of the curve and hence on the degree of the defining polynomial f.

4. Rational functions on curves

Similar to the case of Riemann surfaces, the geometric properties of a smooth projective curve are closely related to the behavior of rational functions on the same surface. For a smooth, projective curve C_f defined by the homogeneous polynomial $f \in \mathbb{F}_q[X,Y,Z]$, we define the *function field* of rational functions by

$$\mathbb{F}_q(C_f) = \left\{ \frac{g(X,Y,Z)}{h(X,Y,Z)} : \deg(g) = \deg(h) \right\} \Big/ \sim \qquad \text{(C5)}$$

with g and h homogenous polynomials in $\mathbb{F}_q[X,Y,Z]$ of identical degree and with equivalence between functions defined by

$$\frac{g}{h} \sim \frac{g'}{h'} \quad \text{if and only if} \quad hg' - gh' \in (f), \qquad \text{(C6)}$$

where (f) is the ideal generated by f. Note that by the requirement that g and h are of the same degree, we have

$$\frac{g(\lambda x,\lambda y,\lambda z)}{h(\lambda x,\lambda y,\lambda z)} = \frac{\lambda^{\deg(g)} g(x,y,z)}{\lambda^{\deg(h)} h(x,y,z)} = \frac{g(x,y,z)}{h(x,y,z)}, \qquad \text{(C7)}$$

which shows that g/h is indeed well defined on the points $(x:y:z)$ in the projective space $\mathbb{P}^2(\mathbb{F}_q)$.

It is an important fact that each nonconstant rational function on C_f has both roots (points where $g=0$) and poles ($h=0$) and that the number of roots equals the number of poles, counting multiplicity (see Sec. IV.H for an example of the structure of $\mathbb{F}_q(C_f)$ for an elliptic curve over $\mathbb{F}_2$).

REFERENCES

Aaronson, S., and A. Ambainis, 2005, Theory Comput. **1**, 47, preliminary version in FOCS 2003.

Adleman, L. M., J. DeMarrais, and M.-D. A. Huang, 1997, SIAM J. Comput. **26**, 1524.

Adleman, L. M., and M.-D. Huang, 2001, J. Symb. Comput. **32**, 171, preliminary version in ANTS-II 1996.

Agrawal, M., N. Kayal, and N. Saxena, 2004, Ann. Math. **160**, 781.

Aharonov, D., 2003, e-print arXiv:quant-ph/0301040.

Aharonov, D., and I. Arad, 2006, e-print arXiv:quant-ph/0605181.

Aharonov, D., I. Arad, E. Eban, and Z. Landau, 2007, e-print arXiv:quant-ph/0702008.

Aharonov, D., and M. Ben-Or, 2008, SIAM J. Comput. **38**, 1207, preliminary version in STOC 1997.

Aharonov, D., V. Jones, and Z. Landau, 2009, Algorithmica **55**, 395, preliminary version in STOC 2006.

Aharonov, D., W. van Dam, J. Kempe, Z. Landau, S. Lloyd, and O. Regev, 2007, SIAM J. Comput. **37**, 166, preliminary version in FOCS 2004.

Ajtai, M., 1996, *ACM Symposium on Theory of Computing* (ACM, New York), pp. 99–108.

Ajtai, M., 1998, *ACM Symposium on Theory of Computing* (ACM, New York), pp. 10–19.

Ajtai, M., and C. Dwork, 1997, *ACM Symposium on Theory of Computing* (ACM, New York), pp. 284–293.

Ajtai, M., R. Kumar, and D. Sivakumar, 2001, *ACM Symposium on Theory of Computing* (ACM, New York), pp. 601–610.

Alagic, G., C. Moore, and A. Russell, 2007, *ACM-SIAM Symposium on Discrete Algorithms* (SIAM, Philadelphia), pp. 1217–1224; e-print arXiv:quant-ph/0603251.

Alexander, J. W., 1923, Proc. Natl. Acad. Sci. U.S.A. **9**, 93.

Ambainis, A., 2007, SIAM J. Comput. **37**, 210, preliminary version in FOCS 2004.

Ambainis, A., A. M. Childs, B. W. Reichardt, R. Špalek, and S. Zhang, 2007, *IEEE Symposium on Foundations of Computer Science* (IEEE, Los Alamitos, CA), pp. 363–372; e-print arXiv:quant-ph/0703015; e-print arXiv:0704.3628.

Ambainis, A., J. Kempe, and A. Rivosh, 2005, *ACM-SIAM Symposium on Discrete Algorithms* (SIAM, Philadelphia), pp. 1099–1108; e-print arXiv:quant-ph/0402107.

Ambainis, A., and R. Špalek, 2006, *Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science Vol. 3884 (Springer, Berlin), pp. 172–183; e-print arXiv:quant-ph/0508205.

Arad, I., and Z. Landau, 2008, e-print arXiv:0805.0040.

Aspuru-Guzik, A., A. D. Dutoi, P. J. Love, and M. Head-Gordon, 2005, Science **309**, 1704.

Babai, L., G. Cooperman, L. Finkelstein, E. Luks, and Á. Seress, 1995, J. Comput. Syst. Sci. **50**, 296, preliminary version in STOC 1991.

Babai, L., D. Grigoriev, and D. Mount, 1982, *ACM Symposium on Theory of Computing* (ACM, New York), pp. 310–324.

Babai, L., W. M. Kantor, and E. Luks, 1983, *IEEE Symposium on Foundations of Computer Science* (IEEE, Los Alamitos, CA), pp. 162–171.

Babai, L., and E. Szemerédi, 1984, *IEEE Symposium on Foundations of Computer Science* (IEEE, Los Alamitos, CA), pp. 229–240.

Bacon, D., 2008, Quantum Inf. Comput. **8**, 438.

Bacon, D., A. M. Childs, and W. van Dam, 2005, *IEEE Symposium on Foundations of Computer Science* (IEEE, Los Alamitos, CA), pp. 469–478; e-print arXiv:quant-ph/0504083.

Bacon, D., A. M. Childs, and W. van Dam, 2006, Chicago J. Theor. Comput. Sci. **2006**, 2.

Barenco, A., A. Ekert, K.-A. Suominen, and P. Törmä, 1996, Phys. Rev. A **54**, 139.

Barnum, H., and E. Knill, 2002, J. Math. Phys. **43**, 2097.

Beals, R., 1997, *ACM Symposium on Theory of Computing* (ACM, New York), pp. 48–53.

Beals, R., H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf, 2001, J. ACM **48**, 778, preliminary version in FOCS 1998.

Bennett, C. H., 1973, IBM J. Res. Dev. **17**, 525.

Bennett, C. H., E. Bernstein, G. Brassard, and U. Vazirani, 1997, SIAM J. Comput. **26**, 1510.

Berndt, B. C., R. J. Evans, and K. S. Williams, 1998, *Gauss and Jacobi Sums* (Wiley, New York).

Bernstein, E., and U. Vazirani, 1993, *ACM Symposium on Theory of Computing* (ACM, New York), pp. 11–20.

Bernstein, E., and U. Vazirani, 1997, SIAM J. Comput. **26**, 1411, preliminary version in STOC 1993.

Beth, T., 1987, Theor. Comput. Sci. **51**, 331.

Blum, A., A. Kalai, and H. Wasserman, 2003, J. ACM **50**, 506, preliminary version in STOC 1999.

Boneh, D., and R. Lipton, 1995, *Advances in Cryptology*, Lecture Notes in Computer Science Vol. 963 (Springer, Berlin), pp. 424–437.

Bordewich, M., M. Freedman, L. Lovász, and D. Welsh, 2005, Combin. Probab. Comput. **14**, 737.

Born, M., and V. Fock, 1928, Z. Phys. **51**, 165.

Boykin, P. O., T. Mor, M. Pulver, V. Roychowdhury, and F. Vatan, 2000, Inf. Process. Lett. **75**, 101.

Brassard, G., P. Høyer, M. Mosca, and A. Tapp, 2002, in *Quantum Computation and Information*, edited by S. J. Lomonaco and H. E. Brandt, AMS Contemporary Mathematics Series

Vol. 305 (AMS, Providence, RI), pp. 53–74; e-print arXiv:quant-ph/0005055.
Brassard, G., P. Høyer, and A. Tapp, 1997, SIGACT News **28**, 14.
Buchmann, J., 1990, in *Séminaire de Théorie des Nombres, Paris 1988–1989*, Progress in Mathematics Vol. 91 (Birkhäuser, Boston), pp. 27–41.
Buchmann, J., 2004, *Introduction to Cryptography*, 2nd ed., Undergraduate Texts in Mathematics (Springer-Verlag, New York).
Buchmann, J. A., and H. C. Williams, 1989, in *Advances in Cryptology*, Lecture Notes in Computer Science Vol. 435 (Springer, Berlin), pp. 335–343.
Buhler, J. P., H. W. Lenstra, Jr., and C. Pomerance, 1993, *The Development of the Number Field Sieve*, Lecture Notes in Mathematics Vol. 1554 (Springer, New York), pp. 50–94.
Buhrman, H., and R. Špalek, 2006, *ACM-SIAM Symposium on Discrete Algorithms* (SIAM, Philadelphia), pp. 880–889; e-print arXiv:quant-ph/0409035.
Cheung, D., D. Maslov, J. Mathew, and D. Pradhan, 2008, in *Proceedings of the Third Workshop on Theory of Quantum Computation, Communication, and Cryptography*, Lecture Notes in Computer Science Vol. 5106 (Springer, Berlin), pp. 96–104.
Cheung, K. K. H., and M. Mosca, 2001, Quantum Inf. Comput. **1**, 26.
Childs, A. M., R. Cleve, E. Deotto, E. Farhi, S. Gutmann, and D. A. Spielman, 2003, *ACM Symposium on Theory of Computing* (ACM, New York), pp. 59–68; e-print arXiv:quant-ph/0209131.
Childs, A. M., and J. Goldstone, 2004a, Phys. Rev. A **70**, 042312.
Childs, A. M., and J. Goldstone, 2004b, Phys. Rev. A **70**, 022314.
Childs, A. M., L. J. Schulman, and U. V. Vazirani, 2007, *IEEE Symposium on Foundations of Computer Science* (IEEE, Los Alamitos, CA), pp. 395–404; e-print arXiv:0705.2784.
Childs, A. M., and P. Wocjan, 2007, Quantum Inf. Comput. **7**, 504.
Childs, A. M., and W. van Dam, 2007, *ACM-SIAM Symposium on Discrete Algorithms* (SIAM, Philadelphia), pp. 1225–1234; e-print arXiv:quant-ph/0507190.
Clausen, M., 1989, Theor. Comput. Sci. **67**, 55.
Cleve, R., 1994, http://www.cpsc.ucalgary.ca/~cleve/pubs/fourier_transform.ps
Cleve, R., 2004, Inf. Comput. **192**, 162, preliminary version in CCC 2000.
Cleve, R., A. Ekert, C. Macchiavello, and M. Mosca, 1998, Proc. R. Soc. London, Ser. A **454**, 339.
Cleve, R., and J. Watrous, 2000, *IEEE Symposium on Foundations of Computer Science* (IEEE, Los Alamitos, CA), pp. 526–536; e-print arXiv:quant-ph/0006004.
Cohen, H., 1993, *A Course in Computational Algebraic Number Theory*, Graduate Texts in Mathematics Vol. 138 (Springer, Berlin).
Coppersmith, D., 1994, IBM Research Division Technical Report No. RC 19642 (unpublished); e-print arXiv:quant-ph/0201067.
Damgård, I. B., 1990, *Advances in Cryptology*, Lecture Notes in Computer Science Vol. 403 (Springer-Verlag, New York), pp. 163–172.
de Beaudrap, J. N., R. Cleve, and J. Watrous, 2002, Algorithmica **34**, 449.
Decker, T., J. Draisma, and P. Wocjan, 2009, Quantum Inf. Comput. **9**, 215.
den Boer, B., 1990, *Advances in Cryptology*, Lecture Notes in Computer Science Vol. 403 (Springer, Berlin), pp. 530–539.
Deutsch, D., 1985, Proc. R. Soc. London, Ser. A **400**, 97.
Deutsch, D., 1989, Proc. R. Soc. London, Ser. A **425**, 73.
Deutsch, D., and R. Jozsa, 1992, Proc. R. Soc. London, Ser. A **439**, 553.
Diaconis, P., 1988, *Group Representations in Probability and Statistics*, IMS Lecture Notes—Monograph Series Vol. 11 (Institute of Mathematical Statistics, Hayward, CA).
Diaconis, P., and D. Rockmore, 1990, J. Am. Math. Soc. **3**, 297.
Diffie, W., and M. E. Hellman, 1976, IEEE Trans. Inf. Theory **22**, 644.
DiVincenzo, D. P., 1995, Phys. Rev. A **51**, 1015.
Dürr, C., M. Heiligman, P. Høyer, and M. Mhalla, 2006, SIAM J. Comput. **35**, 1310, preliminary version in STOC 2000.
Ekert, A., and R. Jozsa, 1996, Rev. Mod. Phys. **68**, 733.
Ettinger, M., and P. Høyer, 1999, e-print arXiv:quant-ph/9901029.
Ettinger, M., and P. Høyer, 2000, Adv. Appl. Math. **25**, 239.
Ettinger, M., P. Høyer, and E. Knill, 1999, e-print arXiv:quant-ph/9901034.
Ettinger, M., P. Høyer, and E. Knill, 2004, Inf. Process. Lett. **91**, 43.
Farhi, E., J. Goldstone, and S. Gutmann, 2008, Theor. Comput. **4**, 169.
Farhi, E., J. Goldstone, S. Gutmann, and M. Sipser, 2000, e-print arXiv:quant-ph/0001106.
Farhi, E., and S. Gutmann, 1998, Phys. Rev. A **58**, 915.
Fenner, S. A., and Y. Zhang, 2008, *Proceedings of the Fifth Annual Conference on Theory and Applications of Models of Computation*, Lecture Notes in Computer Science Vol. 4978 (Springer, Berlin), pp. 70–81, e-print arXiv:quant-ph/0610086.
Feynman, R. P., 1982, Int. J. Theor. Phys. **21**, 467.
Filotti, I. S., and J. N. Mayer, 1980, *ACM Symposium on Theory of Computing* (ACM, New York), pp. 236–243.
Fisher, D. S., 1992, Phys. Rev. Lett. **69**, 534.
Flaxman, A. D., and B. Przydatek, 2005, *Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science Vol. 3404 (Springer, Berlin), pp. 305–314.
Fortnow, L., and J. D. Rogers, 1998, J. Comput. Syst. Sci. **59**, 240, preliminary version in CCC 1998.
Freedman, M., M. Larsen, and Z. Wang, 2002, Commun. Math. Phys. **227**, 605.
Freedman, M. H., A. Kitaev, M. J. Larsen, and Z. Wang, 2003, Bull., New Ser., Am. Math. Soc. **40**, 31.
Freedman, M. H., A. Y. Kitaev, and Z. Wang, 2002, Commun. Math. Phys. **227**, 587.
Friedl, K., G. Ivanyos, F. Magniez, M. Santha, and P. Sen, 2003, *ACM Symposium on Theory of Computing* (ACM, New York), pp. 1–9, e-print arXiv:quant-ph/0211091.
Gavinsky, D., 2004, Quantum Inf. Comput. **4**, 229.
Gordon, D. M., 1993, SIAM J. Discrete Math. **6**, 124.
Grigni, M., L. J. Schulman, M. Vazirani, and U. Vazirani, 2004, Combinatorica **24**, 137, preliminary version in STOC 2001.
Grover, L. K., 1997, Phys. Rev. Lett. **79**, 325, preliminary version in STOC 1996.
Hales, L., and S. Hallgren, 2000, *IEEE Symposium on Foundations of Computer Science* (IEEE, Los Alamitos, CA), pp. 515–525.
Hales, L. R., 2002, Ph.D. thesis, University of California, Berkeley; e-print arXiv:quant-ph/0212002.

Hallgren, S., 2005, *ACM Symposium on Theory of Computing* (ACM, New York), pp. 468–474.
Hallgren, S., 2007, J. ACM **54**, 4, preliminary version in STOC 2002.
Hallgren, S., C. Moore, M. Rötteler, A. Russell, and P. Sen, 2006, *ACM Symposium on Theory of Computing* (ACM, New York), pp. 604–617; e-print arXiv:quant-ph/0511148; e-print arXiv:quant-ph/0511149.
Hallgren, S., A. Russell, and A. Ta-Shma, 2003, SIAM J. Comput. **32**, 916, preliminary version in STOC 2000.
Hamermesh, M., 1989, *Group Theory and Its Application to Physical Problems* (Dover, New York).
Hardy, G. H., and E. M. Wright, 1979, *An Introduction to the Theory of Numbers*, 5th ed. (Oxford University Press, Oxford).
Harrow, A. W., B. Recht, and I. L. Chuang, 2002, J. Math. Phys. **43**, 4445.
Harrow, A. W., and A. Winter, 2006, e-print arXiv:quant-ph/0606131.
Hausladen, P., and W. K. Wootters, 1994, J. Mod. Opt. **41**, 2385.
Hayashi, M., A. Kawachi, and H. Kobayashi, 2008, Quantum Inf. Comput. **8**, 345.
Hoffmann, C. M., 1982, *Group-Theoretic Algorithms and Graph Isomorphism*, Lecture Notes in Computer Science Vol. 136 (Springer-Verlag, Berlin).
Holevo, A. S., 1973, J. Multivariate Anal. **3**, 337.
Høyer, P., 1997, e-print arXiv:quant-ph/9702028.
Hulek, K., 2003, *Elementary Algebraic Geometry*, Student Mathematical Library Vol. 20 (AMS, Providence, RI).
Impagliazzo, R., and A. Wigderson, 1997, *ACM Symposium on Theory of Computing* (ACM, New York), pp. 220–229.
Ip, L., 2003, http://lawrenceip.com/papers/hspsdpabstract.html
Ireland, K., and M. Rosen, 1990, *A Classical Introduction to Modern Number Theory*, Graduate Texts in Mathematics Vol. 84, 2nd ed. (Springer-Verlag, New York).
Ivanyos, G., 2008, Quantum Inf. Comput. **8**, 579.
Ivanyos, G., F. Magniez, and M. Santha, 2003, Int. J. Found. Comput. Sci. **14**, 723, preliminary version in SPAA 2001.
Ivanyos, G., L. Sanselme, and M. Santha, 2007, *Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science Vol. 4393 (Springer, Berlin), pp. 586–597; e-print arXiv:quant-ph/0701235.
Ivanyos, G., L. Sanselme, and M. Santha, 2008, *Latin American Symposium on Theoretical Informatics*, Lecture Notes in Computer Science Vol. 4957 (Springer, Berlin), pp. 759–771; e-print arXiv:0707.1260.
Jaeger, F., D. L. Vertigan, and D. J. A. Welsh, 1990, Math. Proc. Cambridge Philos. Soc. **108**, 35.
Jansen, S., M. B. Ruskai, and R. Seiler, 2007, J. Math. Phys. **48**, 102111.
Jones, V. F. R., 1985, Bull., New Ser., Am. Math. Soc. **12**, 103.
Jordan, S. P., and P. Wocjan, 2008, e-print arXiv:0807.4688.
Jozsa, R., 2003, Ann. Phys. **306**, 241.
Kauffman, L. H., 1987, Topology **26**, 395.
Kawachi, A., T. Koshiba, H. Nishimura, and T. Yamakami, 2005, *Advances in Cryptology*, Lecture Notes in Computer Science Vol. 3494 (Springer, Berlin), pp. 268–284; e-print arXiv:quant-ph/0403069.
Kaye, P., 2005, Quantum Inf. Comput. **5**, 474.
Kaye, P., R. Laflamme, and M. Mosca, 2007, *An Introduction to Quantum Computing* (Oxford University Press, Oxford).
Kedlaya, K. S., 2006, Comput. Complexity **15**, 1.
Khot, S., 2005, J. ACM **52**, 789, preliminary version in FOCS 2004.
Kitaev, A. Y., 1995, e-print arXiv:quant-ph/9511026.
Kitaev, A. Y., 1997, Russ. Math. Surveys **52**, 1191.
Kitaev, A. Y., A. H. Shen, and M. N. Vyalyi, 2002, *Classical and Quantum Computation* (AMS, Providence, RI).
Knill, E., 1995, Los Alamos National Laboratory Technical Report No. LAUR-95–2225 (unpublished); e-print arXiv:quant-ph/9508006.
Knill, E., and R. Laflamme, 1998, Phys. Rev. Lett. **81**, 5672.
Knill, E., R. Laflamme, and W. Zurek, 1996, Los Alamos National Laboratory Technical Report No. LAUR-96–2199 (unpublished); e-print arXiv:quant-ph/9610011.
Knill, E., R. Laflamme, and W. Zurek, 1997, Proc. R. Soc. London, Ser. A **454**, 365.
Köbler, J., U. Schöning, and J. Torán, 1993, *The Graph Isomorphism Problem: Its Structural Complexity* (Springer, New York).
Koblitz, N., 1998, *Algebraic Aspects of Cryptography*, Algorithms and Computation in Mathematics Vol. 3 (Springer-Verlag, New York).
Koiran, P., V. Nesme, and N. Portier, 2007, Theor. Comput. Sci. **380**, 115, preliminary version in ICALP 2005.
Kuperberg, G., 2005, SIAM J. Comput. **35**, 170.
Lauder, A., and D. Wan, 2008, in *Algorithmic Number Theory*, edited by J. P. Buhler and P. Stevenhagen, MSRI Publications Vol. 44 (Cambridge University Press, Cambridge), pp. 579–612.
Lenstra, A. K., H. W. Lenstra, Jr., and L. Lovász, 1982, Math. Ann. **261**, 515.
Lenstra, H. W., Jr., 1983, Math. Op. Res. **8**, 538.
Lenstra, H. W., Jr., 2002, Not. Am. Math. Soc. **49**, 182.
Lidl, R., and H. Niederreiter, 1997, *Finite Fields*, 2nd ed., Encyclopedia of Mathematics and Its Applications Vol. 20 (Cambridge University Press, Cambridge).
Lloyd, S., 1996, Science **273**, 1073.
Lorenzini, D., 1996, *An Invitation to Arithmetic Geometry*, Graduate Studies in Mathematics Vol. 9 (AMS, Providence, RI).
Luks, E. M., 1982, J. Comput. Syst. Sci. **25**, 42.
Magniez, F., and A. Nayak, 2007, Algorithmica **48**, 221, preliminary version in ICALP 2005.
Magniez, F., A. Nayak, J. Roland, and M. Santha, 2007, *ACM Symposium on Theory of Computing* (ACM, New York), pp. 575–584; e-print arXiv:quant-ph/0608026.
Magniez, F., M. Santha, and M. Szegedy, 2007, SIAM J. Comput. **37**, 413.
Manin, Y., 1980, unpublished.
Maslen, D. K., and D. N. Rockmore, 1995, *ACM-SIAM Symposium on Discrete Algorithms* (SIAM, Philadelphia), pp. 253–262.
Maurer, U. M., and S. Wolf, 1999, SIAM J. Comput. **28**, 1689.
Menezes, A. J., P. C. van Oorschot, and S. A. Vanstone, 1996, *Handbook of Applied Cryptography* (CRC, Boca Raton, FL).
Micciancio, D., 2001, SIAM J. Comput. **30**, 2008, preliminary version in FOCS 1998.
Micciancio, D., and S. Goldwasser, 2002, *Complexity of Lattice Problems: A Cryptographic Perspective* (Kluwer, Boston).
Miller, G., 1980, *ACM Symposium on Theory of Computing* (ACM, New York), pp. 225–235.
Miller, G. L., 1976, J. Comput. Syst. Sci. **13**, 300, preliminary version in STOC 1975.
Moore, C., D. Rockmore, and A. Russell, 2006, ACM Trans.

Algorithms **2**, 707, preliminary version in SODA 2004.
Moore, C., D. N. Rockmore, A. Russell, and L. J. Schulman, 2007, SIAM J. Comput. **37**, 938, preliminary version in SODA 2004.
Moore, C., A. Russell, and L. J. Schulman, 2008, SIAM J. Comput. **37**, 1842, preliminary version in FOCS 2005.
Moore, C., A. Russell, and P. Sniady, 2007, *ACM Symposium on Theory of Computing* (ACM, New York), pp. 536–545; e-print arXiv:quant-ph/0612089.
Moore, C., A. Russell, and U. Vazirani, 2007, e-print arXiv:quant-ph/0701115.
Mosca, M., 1999, Ph.D. thesis, University of Oxford.
Mosca, M., and A. Ekert, 1999, *Proceedings of the First NASA International Conference on Quantum Computing and Quantum Communication*, Lecture Notes in Computer Science Vol. 1509 (Springer-Verlag, Berlin).
Mosca, M., and C. Zalka, 2004, Int. J. Quantum Inf. **2**, 91.
Nielsen, M. A., and I. L. Chuang, 2000, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge).
Okamoto, T., K. Tanaka, and S. Uchiyama, 2000, in *Advances in Cryptology*, Lecture Notes in Computer Science Vol. 1880 (Springer, Berlin), pp. 147–165.
Papadimitriou, C. H., 1994, *Computational Complexity* (Addison-Wesley, Reading, MA).
Pérez-García, D., F. Verstraete, M. M. Wolf, and J. I. Cirac, 2007, Quantum Inf. Comput. **7**, 401.
Petrank, E., and M. Roth, 1997, IEEE Trans. Inf. Theory **43**, 1602.
Pila, J., 1990, Math. Comput. **55**, 745.
Pólya, G., 1945, *How to Solve It: A New Aspect of Mathematical Method* (Princeton University Press, Princeton, NJ).
Pomerance, C., 1987, in *Discrete Algorithms and Complexity*, edited by D. S. Johnson, T. Nishizeki, A. Nozaki, and H. S. Wilf (Academic, Boston), pp. 119–143.
Preskill, J., 1998a, Lecture notes for Ph229: Quantum information and computation, http://www.theory.caltech.edu/people/preskill/ph229.
Preskill, J., 1998b, Proc. R. Soc. London, Ser. A **454**, 385.
Proos, J., and C. Zalka, 2003, Quantum Inf. Comput. **3**, 317.
Püschel, M., M. Rötteler, and T. Beth, 1999, *International Symposium on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, Lecture Notes in Computer Science Vol. 1719 (Springer, Berlin), pp. 148–159; e-print arXiv:quant-ph/9807064.
Rabin, M. O., 1980, J. Number Theory **12**, 128.
Radhakrishnan, J., M. Rötteler, and P. Sen, 2009, Algorithmica **55**, 490.
Regev, O., 2004a, J. ACM **51**, 899, preliminary version in STOC 2003.
Regev, O., 2004b, SIAM J. Comput. **33**, 738, preliminary version in FOCS 2002.
Regev, O., 2004c, e-print arXiv:quant-ph/0406151.
Reichardt, B. W., 2004, *ACM Symposium on Theory of Computing* (ACM, New York), pp. 502–510.
Reichardt, B. W., and R. Špalek, 2008, *ACM Symposium on Theory of Computing* (ACM, New York), pp. 103–112; e-print arXiv:0710.2630.
Rivest, R., A. Shamir, and L. Adleman, 1978, Commun. ACM **21**, 120.
Rockmore, D., 1990, Adv. Appl. Math. **11**, 164.
Rudolph, T., and L. Grover, 2002, e-print arXiv:quant-ph/0210187.
Russell, A., and I. E. Shparlinski, 2004, J. Complex. **20**, 404.
Schmidt, A., and U. Vollmer, 2005, *ACM Symposium on Theory of Computing* (ACM, New York), pp. 475–480.
Schnorr, C. P., 1987, Theor. Comput. Sci. **53**, 201.
Schoof, R., 1985, Math. Comput. **44**, 483.
Sen, P., 2006, *IEEE Conference on Computational Complexity* (IEEE, Los Alamitos, CA), pp. 274–287; e-print arXiv:quant-ph/0512085.
Serre, J.-P., 1977, *Linear Representations of Finite Groups*, Graduate Texts in Mathematics Vol. 42 (Springer, New York).
Shenvi, N., J. Kempe, and K. B. Whaley, 2003, Phys. Rev. A **67**, 052307.
Shi, Y., 2003, Quantum Inf. Comput. **3**, 84.
Shor, P. W., 1996, *IEEE Symposium on Foundations of Computer Science* (IEEE, Los Alamitos, CA), pp. 56–65; e-print arXiv:quant-ph/9605011.
Shor, P. W., 1997, SIAM J. Comput. **26**, 1484, preliminary version in FOCS 1994.
Shor, P. W., and S. P. Jordan, 2008, Quantum Inf. Comput. **8**, 681.
Shoup, V., 2005, *A Computational Introduction to Number Theory and Algebra* (Cambridge University Press, Cambridge).
Simon, D. R., 1997, SIAM J. Comput. **26**, 1474, preliminary version in FOCS 1994.
Solovay, R., 2000, Lie groups and quantum circuits, http://www.msri.org/publications/ln/msri/2000/qcomputing/solovay/1/.
Spielman, D. A., 1996, *ACM Symposium on Theory of Computing* (ACM, New York), pp. 576–584.
Stigler, S. M., 1980, Trans. N. Y. Acad. Sci. **39**, 147.
Szegedy, M., 2004, *IEEE Symposium on Foundations of Computer Science* (IEEE, Los Alamitos, CA), pp. 32–41; e-print arXiv:quant-ph/0401053.
Terras, A., 1999, *Fourier Analysis On Finite Groups and Applications*, London Mathematical Society Student Texts Vol. 43 (Cambridge University Press, Cambridge).
Thiel, C., 1995, Ph.D. thesis, Universität des Saarlandes, Saarbrücken, Germany.
van Emde Boas, P., 1981, University of Amsterdam Department of Mathematics Technical Report No. 8104.
van Dam, W., 2002, Algorithmica **34**, 413.
van Dam, W., 2004, e-print arXiv:quant-ph/0405081.
van Dam, W., G. M. D'Ariano, A. Ekert, C. Macchiavello, and M. Mosca, 2007, J. Phys. A **40**, 7971.
van Dam, W., S. Hallgren, and L. Ip, 2006, SIAM J. Comput. **36**, 763.
van Dam, W., M. Mosca, and U. Vazirani, 2001, *IEEE Symposium on Foundations of Computer Science* (IEEE, Los Alamitos, CA), pp. 279–287; e-print arXiv:quant-ph/0206003.
van Dam, W., and G. Seroussi, 2002, e-print arXiv:quant-ph/0207131.
van Dam, W., and U. Vazirani, 2003, http://www.cs.berkeley.edu/~vazirani/pubs/qao.ps
Vollmer, U., 2000, *Proceedings of the Fourth International Symposium on Algorithmic Number Theory*, Lecture Notes in Computer Science Vol. 1838 (Springer, Berlin), pp. 581–594.
von zur Gathen, J., M. Karpinski, and I. Shparlinski, 1997, Comput. Complexity **6**, 64.
Watrous, J., 2001a, *ACM Symposium on Theory of Computing* (ACM, New York), pp. 60–67.
Watrous, J., 2001b, J. Comput. Syst. Sci. **62**, 376.
Watrous, J., 2009, in *Encyclopedia of Complexity and Systems*

Science (Springer, New York); e-print arXiv:0804.3401.
Wiesner, S., 1996, e-print arXiv:quant-ph/9603028.
Witten, E., 1989, Commun. Math. Phys. **121**, 351.
Wocjan, P., and J. Yard, 2008, Quantum Inf. Comput. **8**, 147.
Yao, A. C.-C., 1993, *IEEE Symposium on Foundations of Computer Science* (IEEE, Los Alamitos, CA), pp. 352–361.
Yuen, H. P., R. S. Kennedy, and M. Lax, 1975, IEEE Trans. Inf. Theory **21**, 125.
Zalka, C., 1998, Proc. R. Soc. London, Ser. A **454**, 313.