Quantum Information and Computation, Vol. 22, No. 9&10 (2022) 770–789

AN EXACT QUANTUM HIDDEN SUBGROUP ALGORITHM AND APPLICATIONS TO SOLVABLE GROUPS

MUHAMMAD IMRAN
Department of Algebra, Intitute of Mathematics, Budapest University of Technology and Economics, Műegyetem rkp. 3., Budapest, H-1111, Hungary.

GÁBOR IVANYOS
Institute for Computer Science and Control, Eötvös Loránd Research Network, Kende u. 13-17., Budapest, H-1111, Hungary.

Received February 14, 2022
Revised May 2, 2022

We present a polynomial time exact quantum algorithm for the hidden subgroup problem in $\mathbb{Z}_{m^k}^n$. The algorithm uses the quantum Fourier transform modulo m and does not require factorization of m. For smooth m, i.e., when the prime factors of m are of size $(\log m)^{O(1)}$, the quantum Fourier transform can be exactly computed using the method discovered independently by Cleve and Coppersmith, while for general m, the algorithm of Mosca and Zalka is available. Even for $m = 3$ and $k = 1$ our result appears to be new. We also present applications to compute the structure of abelian and solvable groups whose order has the same (but possibly unknown) prime factors as m. The applications for solvable groups also rely on an exact version of a technique proposed by Watrous for computing the uniform superposition of elements of subgroups.

Keywords: exact quantum algorithm, hidden subgroup problem, abelian group, solvable group

1 Introduction

The first quantum algorithms that demonstrate the advantage of quantum computers over classical ones, such as the procedure proposed by Deutsch and Jozsa [1], are of exact nature, that is, they give the correct solution with probability one. The speedup of these though is rather modest. Later Bernstein and Vazirani provided an oracle problem that can be solved in exact quantum polynomial time, but not in time $n^{O(\log n)}$ on any classical bounded-error probabilistic algorithm [2]. Simon's problem is a simpler one, and it cannot be solved classically in sub-exponential time [3]. However, Simon's polynomial time algorithm may fail with a certain probability (that can though be made arbitrarily small). Shor's celebrated polynomial time quantum algorithms for factoring integers and computing discrete logarithms build on a generalization of the Fourier sampling technique used by Bernstein and Vazirani, and Simon also may fail with a small probability. As observed by several authors, perhaps first by Kitaev [4], Simon's problem and the main ingredients of Shor's algorithms can be cast as instances of the *hidden subgroup problem* (HSP). This is computing a subgroup from a function that is constant on the cosets of the subgroup and takes distinct values on different cosets, see the discussion of the main result for a formal definition. (Note however that

Kitaev's *stabilizer problem* is a slightly less general framework).

As already mentioned, exact quantum algorithms return a correct answer with certainty, while bounded-error methods may give an incorrect answer with probability bounded by a constant less than 1/2, say 1/4. Zero-error algorithms (also called Las Vegas type algorithms) such as Simon's and Shor's methods, are required to either give the correct answer or report failure with probability at most, say, 3/4. Using independent repetition, the error or failure probability can be made arbitrarily small. Thus a polynomial time Las Vegas type quantum algorithm can be interpreted as a correct algorithm that runs in polynomial time in the probabilistic sense and in the unlucky case may take a very long time to succeed, even may (with probability zero) never terminate. In this respect, the relationship between exact and "probabilistic" (bounded-error or zero-error) quantum algorithms is analogous to that between deterministic and randomized classical algorithms. Like for the classical counterpart, investigating the power and limitations of exact quantum algorithms is a challenging problem. Even for various specific algorithmic questions, it would be interesting to know whether exact quantum algorithms have similar advantage over classical ones as probabilistic quantum methods. In this paper we address this question for the the hidden subgroup problem in the additive group $\mathbb{Z}_m^n$ and give an affirmative answer.

To our knowledge the only "interesting" infinite class of abelian groups for which exact polynomial time quantum algorithms for the general hidden subgroup problem have been known is $\mathbb{Z}_2^n$. We consider G interesting if no classical deterministic polynomial time algorithm is known for the hidden subgroup problem in G. When the elementary abelian subgroups of G are either cyclic or of polynomial size and the factorization of the order of G is known then there is such a classical algorithm: a recursion based on determining the intersection of the hidden subgroup with the maximal elementary abelian subgroups. On the other hand, one can adapt Simon's argument [3] to show that the classical randomized query complexity of the HSP in $\mathbb{Z}_p^n$ is $\Omega(p^{\frac{n-1}{2}}.)$ It may be worth mentioning that Ettinger, Hoyer and Knill [5] gave an exact quantum algorithm for the HSP in an arbitrary finite group that uses a polynomial number of queries to the function hiding the subgroup. However, the running time of that algorithm is exponential in general. Note that the hidden subgroup problem in the group $\mathbb{Z}_2^n$ includes Simon's problem as a special case. The first exact polynomial time hidden subgroup algorithm in $\mathbb{Z}_2^n$ is due to Brassard and Hoyer [6]. Their method combines Simon's algorithm with a version of Grover's technique [7] for quantum search. Later Mihara and Sung proposed [8] a simplified version of the Brassard–Hoyer algorithm for the HSP in $\mathbb{Z}_2^n$ based on a slightly different approach. Recently, Bonnetain in [9] devised an improved variant that follows a similar structure. Mosca and Zalka [10] proposed an exact polynomial time solution to the discrete logarithm problem in a cyclic group of known, but not necessarily smooth order m. This problem can be cast as a special instance of the hidden subgroup problem in $\mathbb{Z}_m \times \mathbb{Z}_m$. The method is based on the exact quantum Fourier transform algorithm modulo m proposed in the same paper.

In [6] the authors propose an approach to extend their result to other abelian groups G, e.g., to $G = \mathbb{Z}_m^n$, provided that two problems become efficiently solved. One of them is just the exact computation of the quantum Fourier transform modulo m. At the time of publishing [6], this was known for smooth m, see the manuscripts of Cleve [11] and Coppersmith [12]. The exact quantum Fourier transform proposed by Mosca and Zalka [10] solves this problem for

general m. The other problem is constructing a Boolean function on G with certain properties that support amplitude amplification to obtain new information about the hidden subgroup with certainty. To our knowledge, apart from the case $m = 2$, no such construction has been published although the authors of [6] were planning to investigate the issue for groups of smooth order. In this paper, using ideas from the discrete logarithm algorithm of Mosca and Zalka, we actually solve a modified version of the problem for general m: we construct a function with the desired properties, though on $G \times \{0, 1\}$, rather than on G.

See the remarks at the end of Subsection 3.1 for details.

Models for exact quantum computations. Bernstein and Vazirani define complexity classes related to exact quantum computing (such the class EQP) based on quantum Turing machines in [2]. Unfortunately, in contrast to bounded error quantum computing, the power of this model turns to be quite restricted for the purposes of exact versions of Fourier sampling based approaches. Indeed, as it is pointed out by Nishimura and Ozawa in [13], the quantum Fourier transform QFT_m modulo m can be implemented by a single Turing machine only for a *finite* set of integers m. Therefore, we consider more powerful models based on *quantum circuits*.

We refer the reader to the paper [14] by Nishimura and Ozawa for the details of the definition of uniform quantum circuit families. Here we provide a brief summary. A quantum circuit family is a sequence of circuits built from a given set of elementary gates, called the basis. Uniformity means that the circuit C_s, used for input length s, is required to be constructible by a deterministic Turing machine in time polynomial in the size of C_s including also that the numerical parameters of the gates should be computable by a classical deterministic algorithm in time polynomial in the circuit size and a prescribed precision. Uniform quantum circuit families with finite bases are equivalent to quantum Turing machines at least for polynomial time computations, see [15]. Cleve [11] and Coppersmith [12] noticed that the quantum Fourier transform QFT_m of $\mathbb{Z}_m$ for smooth m (that is, when all the prime factors of m are polynomially small) can be exactly computed by an infinitely based uniform circuit family of polynomial size. Later, Mosca and Zalka [10] proposed a method for implementing QFT_m exactly by a circuit family of size polynomial in $\log m$. That family is uniform in the sense that the parameters of some gates in the circuit for QFT_m can be efficiently (in time $\mathrm{poly}(\log m)$) built knowing m rather than the input size which is essentially $\log m$. This model is apparently stronger than the uniform circuit family in the sense of Nishimura and Ozawa. However, in some applications when m is known to be a member of a set that can be computed in time polynomial from the *size* of the input for the specific application, the Mosca-Zalka implementation of QFT_m can be used as an ingredient of a uniform circuit family in the strict sense. Besides smooth numbers, these cases include numbers m such as the order of the multiplicative group of a finite field of small characteristic or, more generally, the order of the general linear group over such fields.

We remark that Mosca [16] proposed an even more powerful model in which it is allowed to apply a gate U_α with parameter α when (a description of) α emerges *during* the computation. (To be more specific, Mosca proposed allowing gates mapping $|\alpha\rangle|x\rangle$ to $|\alpha\rangle U_\alpha|x\rangle$.) Using that model, he successfully "derandomized" Shor's factoring algorithm. The square-free decomposition algorithm of Li et al. [17] also appears to require application of certain gates with

parameters that depend on the divisors of the number N to factor and hence are not known to be classically computable from the input in polynomial time. (To be more specific, they use QFT_{d-1} to create the uniform superposition of integers between 1 and $d-1$ where d is a "random" divisor of N popping up during the computation. Approaches using amplitude amplification directly could also do the job, but all we are aware of need one or more gates with parameters depending on d.) For the problems considered in this paper, issues with emerging divisors can be circumvented, therefore we only consider the three models discussed above. In increasing order of power, these are

(i) finitely based uniform circuit families;

(ii) infinitely based uniform circuit families; and

(iii) circuit families uniform in the sense that the parameters of the gates are computed from the input.

In (i) and (ii), uniformity is understood in the strict sense, that is, parameters of the gates are computed from the input size.

By a polynomial time quantum algorithm with oracles we mean a family of polynomial size circuits with oracles which, apart from the oracles, is uniform in the strictest sense like model (i), that is, the input size parametrizes the circuits, and the gates are from a finite set. When we do not specify the number of invocations of the oracles in a statement, the number of calls is included in the bound for the circuit size. We state our results in a form when the quantum Fourier transform is accessed via an oracle. This makes it easier to derive the different implications in the three models discussed above. Our algorithms will also be purely unitary, that is, no measurements will be used. The output will be a state in a separate register and will not be entangled with the by-products of the computation. In most cases, the output will be even classical. By that we mean that it is a computational basis state corresponding to a binary string.

The main result. In this paper, we present a polynomial time exact quantum algorithm for the hidden subgroup problem in the additive group $\mathbb{Z}^n_{m^k} = \mathbb{Z}^n/m^k\mathbb{Z}^n$. Our method combines a generalization of the Mihara–Sung variant of the Brassard–Hoyer algorithm (see also Bonnetain's version) with a solution of a variant of the key problem in the approach proposed by Brassard and Hoyer.

The solution is based on some ideas from the discrete logarithm procedure of Mosca and Zalka.

Focusing on groups $\mathbb{Z}^n_{m^k}$ is not too restrictive, as any abelian group of order having the same prime factors as m is a factor of groups of this type. We will consider a slightly generalized version of the HSP which can be dealt with the same Fourier sampling based technique. This generalization captures the application of the *swap test* of Buhrman et al. [18] to testing equality of two quantum states provided that they are either equal or orthogonal. It will also be useful for computations in groups. Watrous in [19] also uses a similar generalization of Shor's order finding to computing orders of group elements modulo normal subgroups. Here is the definition of the generalized HSP.

Let $f : \mathbb{Z}_{m^k}^n \to S(\mathbb{C}^s)$ be a "quantum state valued" function (here $S(\mathbb{C}^s)$ is the unit sphere in $\mathbb{C}^s$). To indicate that the values of f are *states*, we use the notation $|f(x)\rangle$. We say that f hides the subgroup $H \leq \mathbb{Z}_{m^k}^n$ if $|f(x)\rangle = |f(y)\rangle$ whenever $x + H = y + H$ while $|f(x)\rangle$ and $|f(y)\rangle$ are orthogonal if $x + H \neq y + H$. We assume that f is given by an oracle U_f that maps $|x\rangle|0\rangle$ to $|x\rangle|f(x)\rangle$. Here $|0\rangle$ is a fixed unit vector in $\mathbb{C}^r$. (If $s = 2^t$ then the state with all zero qubits is a natural choice.) The task is to compute the hidden subgroup H. Note that in the original (and usual) definition of the hidden subgroup problem, the values of f are from a "classical" set of s elements. From an instance of that, by taking a bijection between the set and a basis of $\mathbb{C}^s$ one easily obtains an instance of the generalized version. In this paper, we omit the adjective "generalized" and use the term HSP for the generalized version. Our main result is the following.

Theorem 1 *There is an exact quantum algorithm that solves the hidden subgroup problem in $\mathbb{Z}_{m^k}^n$ in time $(nk \log m)^{O(1)}$ using $O(nk \log^2 m)$ calls to the oracle U_f or its inverse, and $O(n^2 k \log^2 m)$ applications the quantum Fourier transform QFT_m of $\mathbb{Z}_m$ or its inverse. The procedure outputs the description of the hidden subgroup H by the matrix in Hermite normal form whose columns are a basis of the lattice which is the inverse image of H at the projection map $\mathbb{Z}^n \to \mathbb{Z}^n/m^k\mathbb{Z}^n$. If m is a prime then the number of calls to the oracle U_f or its inverse is $O(nk)$, while the oracle for QFT_m or its inverse is called $O(n^2 k)$ times.*

The numbers m, n and k are assumed to be explicitly given as input, the latter two are in unary. Of course, the most relevant input is provided by the oracle U_f. A unique description of the output is helpful for avoiding that it gets entangled with the garbage. The Hermite normal form appears to be a good choice, as it will also be useful in some of our applications of the hidden subgroup algorithm. For a constant m, e.g., the product of the first 100 primes, the theorem gives an efficient exact solution of the HSP in the strictest model (i) (equivalent to quantum Turing machines). For a smooth m, e.g., when it is input in unary, or it is a product of primes of size polynomial in n and k, the theorem could be used in model (ii), that is in the context of uniform circuits based on an infinite set of gates. For general m, through the implementation of Mosca and Zalka [10], one obtains a result in the strongest model (iii). For $m = 2^\mu - 1$, input by μ in unary, or more generally, when m can be computed in deterministic polynomial time from $[\log m]$ (counted in unary), application of [10] again results in an efficient hidden subgroup algorithm in the (infinitely based) uniform circuit model (ii).

Koiran, Nesme and Portier showed in [20] that the bounded-error quantum query complexity of the hidden subgroup problem in a finite abelian group of rank n (such as the group $\mathbb{Z}_{m^k}^n$) is $\Theta(n)$. Thus, regarding the number of queries, the overhead of our exact method for $\mathbb{Z}_m^n$ is constant factor if m is a prime number and $O(\log^2 m)$ in the general case. For $\mathbb{Z}_{m^k}^n$ the overhead is $O(k \log^2 m)$ (or $O(k)$ when m is prime). In the classical setting, Nayak [21] proposed a simple deterministic polynomial time method that solves the hidden subgroup problem in an abelian group G using $O(\sqrt{|G|})$ queries. The method has an extension to the non-commutative case that uses $O(\sqrt{|G| \log |G|})$ queries. Note that for $G = \mathbb{Z}_2^n$, the classical randomized query complexity of the HSP is shown to be $\Omega(\sqrt{|G|})$ by Simon [3], so in this group Nayak's algorithm has a constant overhead only.

Applications. Watrous [19] gave a polynomial time quantum algorithm for computing the order of a solvable black-box group with unique encoding of elements. Black-box groups are used to obtain general algorithms for various "real" groups such as the multiplicative groups modulo given integers, permutation groups and matrix groups over finite fields. A matrix group over the q-element field $\mathbb{F}_q$ is just a subgroup of the group $GL_n(\mathbb{F}_q)$ of invertible n by n matrices over $\mathbb{F}_q$. Elements of a black-box group are represented by binary strings of a certain length ℓ and the group operations are given by oracles and as input, a generating set for the group is given. See Section 2 for more details.

We use the exact Fourier transform of $\mathbb{Z}_m$ and the hidden subgroup algorithm of $\mathbb{Z}_m^n$ (for various values of n) to devise an exact version of Watrous's algorithm and to solve some other problems for abelian and solvable black-box groups with unique encoding of elements whose order is a divisor of m^k for some k. Our assumption on having the information m about the order is not standard. In fact, knowing a multiple of the order of the multiplicative group modulo the number N would make it possible to factor N in randomized polynomial time. However, there are situations when a multiple of the order of a group can be efficiently computed even in deterministic polynomial time. This is the case for permutation groups and linear groups over finite fields. Watrous uses a classical Monte Carlo method by Babai et al. [22] to construct a series of subgroups that his algorithm builds on. Here we will combine (an exact version of) Watrous's technique with a rather direct method [23] that finds abelian normal subgroups in solvable groups to construct the series from the bottom.

The following theorem summarizes all the applications of the exact quantum hidden subgroup algorithm and the exact version of Watrous's algorithm we obtain.

Theorem 2 *There are exact quantum algorithms running in time* $(r\ell \log m)^{O(1)}$ *using calls to oracles for the group operations, the quantum Fourier transform* QTF_m *of* $\mathbb{Z}_m$ *and to the inverse oracles that, in a black box group* G *with unique encoding of elements by* ℓ*-bit strings given by a list of* r *generators, decide whether* G *is solvable of order dividing* m^k *for some positive integer* k*, and if yes, also perform the following tasks.*

(i) Compute the uniform superposition of elements of G*.*

(ii) Decide membership in G*.*

(iii) Compute the order of G*.*

(iv) Compute the commutator subgroup G' *and the derived series of* G*.*

(v) Given a normal subgroup N *of* G *such that* G/N *is abelian, decompose* G/N *as a direct sum of cyclic groups.*

The uniform superposition in (i) is the element $|G\rangle = \frac{1}{\sqrt{|G|}} \sum_{x \in G} |x\rangle$ of the group algebra $\mathbb{C}G \subseteq \mathbb{C}^{2^\ell}$.

We remark that Luks in [23] presents classical deterministic polynomial time algorithms for testing solvability of finite *matrix groups* and for many tasks like (iv) in these groups. However, it is not known whether such classical deterministic algorithms exist for *black-box* groups, even if oracles for order finding and membership test in abelian subgroups are allowed.

Of course, the full machinery of the proof of Theorem 2 is not required for the result in (v) for $N = \{1_G\}$, that is computing the structure of G when G is abelian of order having the same

prime factors as m. In fact, this is a direct consequence of Theorem 1. Among others, the result captures finding orders of group elements, in particular in the multiplicative group $\mathbb{F}_q^*$ of the q-element field $\mathbb{F}_q$. This can be used to find primitive elements in certain fields. (Recall that a primitive element of $\mathbb{F}_q$ is a field element of multiplicative order $q-1$.) When q is a power of the prime p, a subset of $\mathbb{F}_q$ containing at least one primitive element can be constructed deterministically in time $\mathrm{poly}(\log q)$ assuming the extended Riemann hypothesis (ERH) when $q = p$ by the results of Wang and Bach [24, 25] and when $q = p^2$ by Shoup [26]. Shoup [26] and Shparlinski [27] also proposed methods that work unconditionally in time $p\,\mathrm{poly}(\log q)$. Not much later Perel'muter and Shparlinski [28] came up with a construction that requires time $\sqrt{p}\,\mathrm{poly}(\log q)$, see also Lemma 7 in [29] for a more explicit bound. Combining these with order finding, one immediately obtains the following.

Corollary 1 *Let q be a power of the prime p. Then a primitive element in $\mathbb{F}_q$ can be found by an exact quantum algorithm using QFT_{q-1} and its inverse in time $\sqrt{p}\,\mathrm{poly}(\log q)$. Furthermore, if $q = p$ or $q = p^2$ then, assuming ERH, a primitive element in $\mathbb{F}_q$ can be found by an exact quantum algorithm using QFT_{q-1} and its inverse in time* $\mathrm{poly}(\log q)$.

We remark that for $p = \mathrm{poly}(\log q)$, the first part of this corollary implies existence of an exact polynomial time quantum algorithm in model (ii).

The structure of the paper is the following. In Section 2 we recall further definitions, discuss the notation and the terminology used in this work. Some standard facts and techniques are also recalled there. Section 3 is devoted to the exact hidden subgroup algorithm, while the applications (mostly in abelian and solvable groups) are discussed in Section 4.

2 Notation, terminology and preliminaries

This section is devoted to definitions not yet given in the Introduction, to introducing notation and terminology that are less known or not standard and to discussion of some known facts and techniques that serve as ingredients of the algorithms presented in this paper.

It is common to define exact quantum procedures as those involving measurements that produce the desired result with probability one. By delaying the measurements until the end and omitting the final measurement, one obtains an equivalent unitary procedure. As already mentioned, in this paper we consider the unitary version. With a few exceptions such as the quantum Fourier transform, we require (or ensure) that an exact quantum procedure is a circuit that implements a unitary transformation that maps $|x\rangle|0\rangle|0\rangle$ to $|x\rangle|output(x)\rangle|garbage(x)\rangle$ where $|x\rangle$ is the computational basis state corresponding to the input string x and $|output(x)\rangle$ is the state which is the output corresponding to the actual input string x while $|garbage(x)\rangle$ contains the by-products of the computations. From the initial state $|x\rangle|0\rangle|0\rangle$ we will frequently omit the third register as well as in some cases even the second one. Similarly, during the description of procedures some registers not mentioned previously may pop up, meaning a piece of memory initialized to the zero string of appropriate size. Also, when the content of a temporary storage is reset to zero, then it can be left away from the rest of the description. The output is often classical. By that we mean that $|output(x)\rangle$ is a computational basis state corresponding to the string describing the actual output. In that case, the garbage can be cleared by standard techniques.

For standard notions from group theory such as subgroups, normal subgroups, commutator or derived subgroup, derived series, solvability, etc., we refer the reader to the textbooks, e.g.,

to [30]. A finite solvable group G has a subnormal series with cyclic factors, that is, a sequence of subgroups $\{1\} = G_0 < G_1 < \ldots < G_h = G$ such that $G_i \lhd G_{i+1}$ and G_{i+1}/G_i is cyclic for $0 \leq i < h$. We will use the term *polycyclic series* for such sequences. A polycyclic series is usually given by a list of group elements $g_1, \ldots, g_h$ such that G_i is a generated by $g_1, \ldots, g_i$ $(i = 1, \ldots, h)$.

The concept of black-box groups was introduced by Babai and Szemerédi [31] for studying the structure of finite matrix groups. As already mentioned, elements of a black-box group (with unique encoding) are represented by binary strings of a certain length ℓ and the group operations (multiplication, taking the identity element and taking inverses) are given by oracles. In the quantum setting, the oracle for multiplication is assumed to be a unitary transformation of $\mathbb{C}^{2^{\ell \cdot 3}}$ mapping $|x\rangle|y\rangle|0\rangle$ to $|x\rangle|y\rangle|z\rangle$ where z encodes the product of the group elements represented by x and y whenever they both encode valid group elements. As we can easily find a multiple of the order of a given element, we do not actually need the oracles for taking the identity element and inverses. (Given x, a multiple of the order can be the smallest number of the form m^t with $1 \leq t \leq \ell$ such that $x^{m^t+1} = x$ and then $1_G = x^{m^t}$ and $x^{-1} = x^{m^t-1}$.) As input, a generating set for the group is given. We remark that in computational group theory, to capture factor groups by certain normal subgroups, it is common to also consider black-box groups with non-unique encoding of elements. In that case, several code-words can represent the same group element and a further oracle is included to test equality. As computing the structure of already an elementary abelian black-box group with non-unique encoding becomes difficult even on quantum computers, see [32], we do not consider this general concept in this paper.

2.1 Subgroups of $\mathbb{Z}_{m^k}^n$, lattices and normal form matrices

We will represent a subgroup A of $\mathbb{Z}_{m^k}^n$ by a special basis for the lattice L_A in $\mathbb{Z}^n$ which is the inverse image of A at the projection map $\mathbb{Z}^n \to \mathbb{Z}^n/m^k\mathbb{Z}^n = \mathbb{Z}_{m^k}^n$. The lattice L_A contains $m^k\mathbb{Z}^n$ therefore it has full rank n.

Recall that for every n by s integer matrix M there exists an s by s unimodular matrix (an integer matrix with determinant ± 1) U such that $H = MU$ where

1. H is lower triangular whose full zero columns are located to the right of any other column.
2. The nonzero diagonal entries of H are positive.
3. The off-diagonal elements of H are non-negative and strictly less then the diagonal element of their rows.

H is called the *Hermite normal form* of M. The Hermite normal form H is uniquely determined by the lattice L_M in $\mathbb{Z}^n$ spanned by the columns of M therefore its nonzero columns form a well-defined basis of L_M. We will use the n by n matrix H_A in Hermite normal form to represent the lattice L_A. For example, if A is the trivial subgroup of $\mathbb{Z}_{m^k}^n$ then $L_A = m^k\mathbb{Z}^n$ and H_A is the n by n scalar matrix $m^k \cdot \mathrm{Id}$.

Also, there exist unimodular matrices L and R of sizes n by n and s by s, respectively, such that the matrix $S = LMR$, called the *Smith normal form* of M, whose positive entries $d_1, \ldots, d_r$ are located in the first r columns (equivalently, rows) with $d_i | d_{i+1}$ for $1 \leq i < r$. The Smith normal form is also uniquely determined by L_M.

The unimodular matrices U, R and L are referred to as *multipliers*. The right multipliers U and R correspond to changing the basis for L_M while the left multiplier L corresponds to changing the basis for $\mathbb{Z}^n$. The normal form matrices H and S as well as the multipliers can be computed in deterministic polynomial time (in the number of bits representing M), see e.g., [33].

Besides providing unique representations of subgroups, the Hermite normal form can be used to solve certain subtasks efficiently. For example, we will work with pairs of subgroups A, B of $\mathbb{Z}^n_{m^k}$ for which we already know that $A \leq B$ and need to decide whether they are equal. Then this can be readily tested by comparing the Hermite normal form matrices H_A and H_B representing these subgroups. Also, if $A < B$ then the first column of H_B whose diagonal entry is smaller than the corresponding entry in H_A will be an element of $B \setminus A$.

The Smith normal form together with the multipliers can be used to solve systems of linear congruences such as the following efficiently. We consider the standard scalar product $(x, y) = x^T y$ modulo m. For $A \leq \mathbb{Z}^n_m$ the subgroup $A^\perp$ consists of the elements $y \in \mathbb{Z}^n_m$ such that $(x, y) = 0$ for every $x \in A$. We have $L_{A^\perp} = \{z \in \mathbb{Z}^n : H_A^T z \in m\mathbb{Z}^n\}$. Now if $S = LH_AR$ is the Smith normal form of H_A then substituting $z' = (L^T)^{-1}z$ we obtain $(L^T)^{-1}L_{A^\perp} = \{z' \in \mathbb{Z}^n : S^T z' \in m\mathbb{Z}^n\}$. A basis of this lattice can be easily obtained, as S is diagonal. A basis for $L_{A^\perp}$ can be obtained by multiplying by L^T. A further Hermite normal form calculation gives the unique representation of $A^\perp$.

Recall that any abelian group A generated by n elements can be presented as a factor of a free abelian group $\mathbb{Z}^n$. If the generators $a_1, \ldots, a_n$ are fixed, then elements of the kernel of the map $\phi : (x_1, \ldots, x_n)^T \mapsto \prod a_i^{x_i}$ are the *relations* corresponding to the generators. It is usual to write a relation $(x_1, \ldots, x_n)^T$ in the form $\prod a_i^{x_i}$. A set of *defining relations* is any collection that generate the kernel. An Hermite normal form matrix for the kernel is a sort of standard and rather economical list of defining relations for the fixed set of generators. Theorem 1 will be applicable in the special cases when $a_i^{m^k} = 1_A$ for $i = 1, \ldots, n$. Then the kernel of ϕ contains $m^k\mathbb{Z}^n$ and hence ϕ induces a homomorphism f from $\mathbb{Z}^n_{m^k}$ onto A. Let H be the subgroup hidden by f. Then the kernel of ϕ is the lattice L_H. Therefore, the output of the algorithm of Theorem 1 is just the Hermite normal form matrix for the relations. If A is cyclic and $n = 1$ then this 1 by 1 matrix contains the order of u_1. More generally, the order of A is the product of the diagonal entries of the corresponding Hermite normal form matrix. Perhaps it is worth mentioning the case when $n = 2$ and A is the cyclic group generated by u_2. Then the Hermite normal form matrix will be

$$\begin{pmatrix} 1 & 0 \\ o_2 - d & o_2 \end{pmatrix},$$

where o_2 is the order of u_2 and d is the base-u_2 discrete logarithm of u_1.

By changing the generators, one may obtain possibly even better and even more economical presentations. In particular, A is isomorphic to the direct sum $\mathbb{Z}_{m_1} \oplus \ldots \oplus \mathbb{Z}_{m_{n'}}$ with $n' \leq n$, $m_i > 1$, and $m_{i-1}|m_i$ for $1 < i$. That is, there exist generators $b_1, \ldots, b_{n'}$ for A with "diagonal" defining relations $b_1^{m_1}, b_2^{m_2}, \ldots, b_{n'}^{m_{n'}}$. Such generators and relations can be found using the Smith normal form of an initial matrix of defining relations. In our special case, the Smith normal form will be the diagonal matrix $\mathrm{diag}(1, \ldots, 1, m_1, \ldots, m_{n'})$ and the left multiplier gives expressions for the new generators in terms of the original ones.

2.2 Amplitude amplification from $1/\sqrt{2}$ to 1

Brassard and Hoyer in [6] proposed a method, based on a technique similar to the rotation used in Grover's search [7], to get rid of the "undesirable half" of a quantum state provided that the state can be produced by a unitary procedure. We state here a version of their result as follows.

Assume that there is a unitary procedure $\mathcal{U}$ acting on s qubits that maps the state $|0\rangle$ to $c_0|B\rangle|0\rangle + c_1|A\rangle|1\rangle$, where $|A\rangle$ and $|B\rangle$ are of unit norm and the last register contains just one qubit. Then there is a unitary procedure $\mathcal{U}'$ that, using $\mathcal{U}$ or its inverse 3 times and $O(s)$ ordinary gates, maps the state $|0\rangle$ to $c_0'|B\rangle|0\rangle + c_1'|A\rangle|1\rangle$ for some complex numbers c_0' and c_1' such that in the case when $c_0 = c_1 = \frac{1}{\sqrt{2}}$ then $c_0' = 0$ and $|c_1'| = 1$, that is, the outcome is (up to a phase) $|A\rangle|1\rangle$. Also, if $c_1 = 0$ then $c_1' = 0$, i.e., in that case $\mathcal{U}'$ gives $|B\rangle|0\rangle$ (up to a phase). The result can be easily extended to the case when $\mathcal{U}$ maps $|x\rangle|0\rangle$ to $\frac{1}{\sqrt{2}}|x\rangle|B_x\rangle|0\rangle + \frac{1}{\sqrt{2}}|x\rangle|A_x\rangle|1\rangle$. That is, $\mathcal{U}$ may have an input that is left intact by $\mathcal{U}$.

We will apply this in our hidden subgroup algorithm (Section 3). There we use it in a form closer to that in [6]. We assume availability of a procedure $\mathcal{U}_0$ that produces a state $\sum_{y\in S} c_y|y\rangle|\gamma_y\rangle$ where S is a set of binary strings of a certain length, $|y\rangle$ denotes the computational basis vector corresponding to y and $|\gamma_y\rangle$ is of unit norm. We also suppose that we have a unitary procedure $\mathcal{U}_1$ that maps $|y\rangle|0\rangle$ to $|y\rangle|f(y)\rangle$ for some function $f : S \to \{0,1\}$ such that $\sum_{y\in S:f(y)=1} |c_y|^2 = \frac{1}{2}$. Then $\mathcal{U}$ is just the composition $\mathrm{Id}\otimes\mathcal{U}_1\circ\mathcal{U}_0\otimes\mathrm{Id}$ and the "desired half" of the state is $|A\rangle|1\rangle = \sqrt{2}\sum_{y:f(y)=1} c_y|y\rangle|\gamma_y\rangle|1\rangle$. We even apply this in certain cases when $\sum_{y\in S:f(y)=1} |c_y|^2 > \frac{1}{2}$. Then we use the idea of Mosca and Zalka applied in [10]: by modifying f we throw away some of the good y's and adjust the coefficient c_y of some others to obtain a superposition of "total squared amplitude" $\frac{1}{2}$ consisting only of desirable y's.

2.3 The quantum Fourier transform and Fourier sampling

The elements of the group $\mathbb{Z}_m$ are represented by integers between 0 and $m-1$ (in binary). The quantum Fourier transform QFT_m of the group $\mathbb{Z}_m = \mathbb{Z}/m\mathbb{Z}$ maps the computational basis state $|x\rangle$ to the state $\frac{1}{\sqrt{m}}\sum_{y\in\mathbb{Z}_m}\omega^{xy}|y\rangle$, where $\omega = e^{\frac{2\pi\iota}{m}}$ and $(\,,\,)$ stands for the usual scalar product on $\mathbb{Z}_m^n$. Note that the quantum Fourier transform of $\mathbb{Z}_m^n$ can be implemented as the tensor product of n copies of QFT_m.

Fourier sampling involves the first few steps of the known hidden subgroup algorithms. Since we will need some related notation and since we apply it for the *generalized* version of the HSP, and also in the exact variant of Mosca's algorithm, it will be useful to recall details of that standard procedure in our context.

The procedure works on two registers. The first one is for holding group elements from $\mathbb{Z}_m^n$ while the second one is for the states which are the values of f. The first step uses the exact quantum Fourier transform of $\mathbb{Z}_m^n$ to map the state $|0\rangle|0\rangle$ to

$$\frac{1}{m^{n/2}}\sum_{x\in\mathbb{Z}_m^n}|x\rangle|0\rangle.$$

Then an application of the oracle U_f gives

$$\frac{1}{m^{n/2}}\sum_{x\in\mathbb{Z}_m^n}|x\rangle|f(x)\rangle.$$

Finally, another application of the quantum Fourier transform results in

$$|\Psi\rangle = \frac{1}{m^n} \sum_{x,y \in \mathbb{Z}_m^n} \omega^{(x,y)} |y\rangle |f(x)\rangle. \tag{1}$$

Let C be a cross-section of H in $\mathbb{Z}_m^n$. Then every $x \in \mathbb{Z}_m^n$ can uniquely be written as $x = x' + c$ with $x' \in H$ and $c \in C$. It follows that

$$|\Psi\rangle = \frac{1}{m^n} \sum_{y \in \mathbb{Z}_m^n} \sum_{c \in C} \sum_{x \in H} \omega^{(x+c,y)} |y\rangle |f(c)\rangle.$$

For fixed $c \in C$ and $y \in \mathbb{Z}_m^n$, we have

$$\sum_{x \in H} \omega^{(x+c,y)} = \begin{cases} \omega^{(c,y)} |H| & \text{if } y \in H^\perp, \\ 0 & \text{otherwise.} \end{cases}$$

(For a proof, notice that map $x \mapsto \omega^{(x,y)}$ is a character χ of H which is the trivial character 1 if and only if $y \in H^\perp$ and use the orthogonality relation for χ and 1.) Therefore, using also that $|H| \cdot |H^\perp| = |H| \cdot |C| = m^n$, we have

$$|\Psi\rangle = \frac{1}{\sqrt{|H^\perp|}} \sum_{y \in H^\perp} |y\rangle |\gamma_y\rangle, \tag{2}$$

where

$$|\gamma_y\rangle = \frac{1}{\sqrt{|C|}} \sum_{c \in C} \omega^{(c,y)} |f(c)\rangle. \tag{3}$$

In our hidden subgroup algorithm, only the first register of $|\Psi\rangle$ (that is, $|y\rangle$) will be used and $|\gamma_y\rangle$ will be considered as garbage. However, $|\gamma_y\rangle$ plays a crucial role in Watrous's algorithm and its exact version, see Subsection 4.2.

3 The exact hidden subgroup algorithm

In this section we prove Theorem 1. We begin with the special case $k = 1$ and conclude the proof with a reduction from the general case to that.

3.1 The HSP in $\mathbb{Z}_m^n$

In this subsection we provide the construction of an exact algorithm for finding the hidden subgroup H of $\mathbb{Z}_m^n$ which requires $O(n \log^2 m)$ queries. We denote by $\mathcal{P}$ the procedure described in Subsection 2.3. Recall that $\mathcal{P}$ maps $|0\rangle|0\rangle$ to the state $|\Psi\rangle$ given in (2).

We use an iteration to compute H. During the iteration, we maintain a subgroup K of H as well as a subgroup L of $H^\perp$. We use the Hermite normal form matrices as described in Subsection 2.1 for representing K and L. Initially $K = \{0\}$ and $L = \{0\}$ and in each round we enlarge either K or L until K becomes equal to $L^\perp$. (Note that the conditions on K and L imply that $K \leq L^\perp$ and that $K = H$ if and only if $K = L^\perp$.)

If $K < L^\perp$ we choose an element $u \in L^\perp \setminus K$. Computing $L^\perp$, testing equality of K and $L^\perp$, and in the case when $K < L^\perp$ finding an u from the difference can be easily done using the Hermite and Smith normal form methods described in Subsection 2.1.

The map $\mu_u : \mathbb{Z}_m^n \to \mathbb{Z}_m$ defined as $\mu_u(y) = (u, y)$ is a homomorphism from $\mathbb{Z}_m^n$ to $\mathbb{Z}_m$. Therefore, the image $\mu_u(H^\perp)$ is a subgroup of $\mathbb{Z}_m$ and for each $a \in \mu_u(H^\perp)$ the number of elements $y \in H^\perp$ such that $\mu_u(y) = a$ is $\frac{|H^\perp|}{|\mu_u(H^\perp)|}$. Notice that $\mu_u(H^\perp)$ is the trivial subgroup of $\mathbb{Z}_m$ if and only if $u \in H^{\perp\perp} = H$. Our aim is to find an element $y \in H^\perp$ such that (u, y) is nonzero if (and only if) $u \notin H$. To be more specific, for $u \notin H$ we want to "distill" a superposition of certain states $|y\rangle|\gamma_y\rangle$ with $(u, y) \neq 0$. To this end, we use the amplitude amplification technique of Brassard and Hoyer [6] described in Subsection 2.2, combined with the idea of Mosca and Zalka [10] to tailor the "total squared amplitude" to $\frac{1}{2}$.

Assume that $u \notin H$. Let d be the smallest positive integer such that $d + m\mathbb{Z} \in \mu_u(H^\perp)$. Then d is a divisor of m and the nonzero elements of $\mu_u(H^\perp)$ are represented by the $\frac{m}{d} - 1$ positive integers of the form td with $td < m$. Also, if $\frac{m}{d}$ is even then the integers of the form td such that $m/2 \leq td < m$ represent just half of the elements of $\mu_u(H^\perp)$. However, if $\frac{m}{d}$ is odd then to get the desired "half", we need to add a further element of $\mu_u(H^\perp)$, say d, with weight $\frac{1}{2}$. The point is that we do not know d. However, fortunately, for at least one integer $0 \leq j \leq \log_2 m$, namely for $j = \lceil \log_2 d \rceil$, the interval $(0, 2^j]$ contains only d and no other multiple of d. (Indeed, if $j - 1 < \log_2 d \leq j$ then $d \leq 2^j$ and $2d > 2^j$.)

Based on the discussions above, we have the following exact quantum algorithm for hidden subgroup problem in $\mathbb{Z}_m^n$:

Algorithm 1 Exact Quantum Algorithm for HSP in $\mathbb{Z}_m^n$

Initialize: $K \leftarrow \{0\}$ and $L \leftarrow \{0\}$;
while $K \neq L^\perp$ **do**
  Take $u \in L^\perp \setminus K$;
  $Found \leftarrow$ False;
  for $j = -1, \ldots, \lfloor \log_2 m \rfloor$ **do**
    ▷ $f(j, x, b) = \begin{cases} 1 & \text{if } (u, x) \geq \frac{m}{2} \text{ or } b = 1 \text{ and } 0 < (u, x) \leq 2^j \\ 0 & \text{otherwise} \end{cases}$
    ▷ $\mathcal{U}_j : |0\rangle|0\rangle|0\rangle|0\rangle \mapsto |\psi\rangle = \frac{1}{\sqrt{2|H^\perp|}} \sum |x\rangle|\gamma_x\rangle|b\rangle|f(j, x, b)\rangle$,
    ▷ where the summation goes through all $x \in H^\perp$, $b \in \{0, 1\}$.
    Apply the amplitude amplified version of $\mathcal{U}_j$;
    ▷ Obtain $|\psi'_j\rangle = \sum c'_{f(j,x,b)}|x\rangle|\gamma_x\rangle|b\rangle|f(j, x, b)\rangle$,
    Look at the $|x\rangle$-register;
    if $(u, x) \neq 0$ **then**
      $Found \leftarrow$ True;
      $L \leftarrow \langle L \cup \{x\}\rangle$;
    end if
  end for
  if $Found =$ False **then**
    $K \leftarrow \langle K \cup \{u\}\rangle$;
  end if
end while

Each round consists of iterations for $j = -1, \ldots, \lfloor \log_2 m \rfloor$ of the following procedure. Like Mosca and Zalka [10], we attach a one-qubit register and, in addition to calling the procedure $\mathcal{P}$, we also apply a one-qubit Hadamard transform to get the qubit $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. We use the Brassard-Hoyer amplitude amplification (see Subsection 2.2), where we accept $|x\rangle|b\rangle$ (compute the qubit $|1\rangle$ in a further register) iff either $\mu_u(x) \geq m/2$ or $b = 1$ and $0 < \mu_u(x) \leq 2^j$. The case when $\frac{m}{d}$ is even is covered at least once, when $j = -1$ where the interval $(0, 2^j] = (0, \frac{1}{2}]$ does not contain any integer. While the case when $\frac{m}{d}$ is odd is covered at least once, when $j = \lceil \log_2 d \rceil$. If $\mu_u(H^\perp)$ is trivial then for each j we get x with $\mu_u(x) = 0$, else for at least one j we get an $x \in H^\perp$ such that $\mu_u(x)$ is nonzero. In the former case, we have $u \in H \setminus K$ and

we can replace K with the subgroup generated by K and u. In the latter case, $x \in H^\perp \setminus L$ and, we can enlarge L by replacing it with the subgroup generated by L and x.

As in each round before termination, the product of the sizes of the subgroups K and L is increased by at least a factor 2, we need at most $\lceil \log_2 |\mathbb{Z}_m^n| \rceil = \lceil n \log_2 m \rceil$ rounds of iterations. The overall number of calls to procedure $\mathcal{P}$ or its inverse and through those the number of calls to U_f or its inverse as well as the number of applications of the Fourier transform of $\mathbb{Z}_m^n$ or its inverse is $O(n \log^2 m)$. Thus, QFT_m or its inverse is applied $O(n^2 \log^2 m)$ times. The required number of elementary gates is $(n \log m)^{O(1)}$.

Remarks. (1) When m is an odd prime and $u \notin H$ then we have $d = 1$ therefore considering the case $j = 0$ only is sufficient. Also, in each step K or L gets enlarged by a factor p. Therefore, for a prime m we have an algorithm using $O(n)$ queries.
(2) The key problem in the approach proposed by Brassard and Hoyer [6] to extend their method from $\mathbb{Z}_2^m$ to $\mathbb{Z}_m^n$ is existence and efficient computability of a Boolean function $\lambda : \mathbb{Z}_m^n \to \{0,1\}$ such that λ is identically zero on the subgroup what is denoted by L in our context while it takes value one on exactly half (or another, prescribed sufficiently large fraction) of the elements of the (unknown) $H^\perp$ unless $K = H^\perp$. Using that, one can use the method of Subsection 2.2 to obtain an element of $H^\perp \setminus K$ or conclude $K^\perp = H$ with certainty. Notice that if $u \notin H$ then the function $f(j, \cdot, \cdot) : \mathbb{Z}_m^n \times \{0,1\}$ defined in Line 6 of the algorithm has analogous properties for the appropriate j: it is identically zero on $K \times \{0,1\}$ while nonzero on exactly the half of the pairs from $H^\perp \times \{0,1\}$. Thus, method for tailoring the "total squared amplitude" can be considered as a solution of a modified version of the problem of Brassard and Hoyer. Note that amplitude amplification using $f(j, \cdot, \cdot)$ may happen to find an element of $H^\perp \setminus L$ not only for the appropriate j. The current pseudo-code adds a new element found by the largest "successful" j.

In fact, at this point of the algorithm it cannot be determined in general which one of the functions $f(j, \cdot, \cdot)$ takes nonzero value exactly on the half of $H^\perp$.

3.2 Extending to the HSP in $\mathbb{Z}_{m^k}^n$

We describe how to reduce the hidden subgroup problem of $\mathbb{Z}_{m^k}^n$ to that of $\mathbb{Z}_m^n$. Assume that f hides the subgroup H. We will use an iteration during which we maintain a subgroup H_0 of H. In each round H_0 is increased until we conclude that $H = H_0$. Initially $H_0 = \{0\}$. It will be convenient to represent H_0 by a matrix in Smith normal form (together with multipliers, in particular the left multiplier for the basis change of $\mathbb{Z}^n$) whose columns are a basis for the lattice L_{H_0}.

We start each round with computing the subgroup $K_0 = \{x \in \mathbb{Z}_{m^k}^n : mx \in H_0\}$. This can be efficiently done using the Smith normal form representation of H_0. (Indeed, if the Smith normal form matrix for H_0 is $\mathrm{diag}(d_1, \ldots, d_n)$ the matrix for K_0 is $\mathrm{diag}(\frac{d_1}{\gcd(d_1,m)}, \ldots, \frac{d_n}{\gcd(d_n,m)})$. When $K_0 = H_0$ we can stop as in that case H cannot be larger than H_0. Otherwise, we consider the factor group $K = K_0/H_0$. The group K is isomorphic to $\bigoplus_{i=1}^n \mathbb{Z}_{\gcd(d_i,m)}$ and one can find a map $\phi_0 : \mathbb{Z}_m^n \to K_0$ such that the map $x \mapsto \phi_0(x) + H_0$ is a homomorphism from $\mathbb{Z}_m^n$ onto K. Using the simultaneous diagonal form for H_0 and K_0 such a ϕ_0 can be defined by mapping $(x_1, \ldots, x_m)^T$ to $(x'_1 \frac{d_1}{\gcd(d_1,m)}, \ldots, x'_n \frac{d_n}{\gcd(d_n,m)})^T$, where x'_i is the least *positive* integer congruent with x_i modulo $\gcd(d_i, m)$. The composition $f \circ \phi_0$ defines a function hiding

a subgroup S of $\mathbb{Z}_m^n$ such that the subgroup generated at $\phi_0(S)$ and H_0 generates $K_0 \cap H$. We compute generators for S using the hidden subgroup algorithm for $\mathbb{Z}_m^n$. The images of the generators at ϕ_0 and generators for H_0 will generate $K_0 \cap H$. We compute the Smith normal form matrix representing $K_0 \cap H$ and test if this subgroup equals H_0. If yes, we can stop because in that case $H = H_0$. Otherwise, we replace H_0 with $K_0 \cap H$ and proceed with the next round. Using an induction on j, one can show that H_0 after the jth round contains $H \cap m^{k-j}\mathbb{Z}_{m^k}^n$ $(j \leq k)$. Therefore, the procedure requires at most k rounds.

4 Applications to groups

In this section we prove Theorem 2. We start with some rather direct applications of the exact hidden subgroup algorithm (Subsection 4.1). For applicability in the proof of Theorem 2, some of the statements are rather technical, they include the assumption of availability of a procedure that computes the uniform superposition $|K\rangle$ of the element of a subgroup K. Of course, when K is the trivial subgroup $\{1_G\}$ then this is easy as $|K\rangle = |1_G\rangle$. For this K we obtain results for abelian groups that may be of interest on their own right. Throughout the section we assume that G is a black box group with unique encoding of elements with binary strings of length ℓ.

4.1 Some basic applications

4.1.1 An exact version of the swap test

The swap test can be used to decide equality of two quantum states, provided that they are either equal (up to a phase) or orthogonal to each other. In the usual setting, the states are given as input and orthogonality is detected with probability $\frac{1}{2}$. This suggests that the amplitude amplification technique of Brassard and Hoyer (see Subsection 2.2) can be used to get an exact version. However, availability of the two states as input is not sufficient for applying the technique. It rather requires *procedures for creating* the two states. In that setting, one could directly combine the swap test with the amplitude amplification to obtain an exact procedure. Interestingly, the idea of swapping the two states conditionally suggests an interpretation as an instance of (the generalized version of) the hidden subgroup problem in the two-element group $\mathbb{Z}_2$.

Corollary 2 *Assume that there are two unitary procedures $\mathcal{P}_1$ and $\mathcal{P}_2$ mapping the ℓ-qubit state $|0\rangle$ to $|\psi_1\rangle$ and to $|\psi_2\rangle$, respectively. Suppose further that the states $|\psi_1\rangle$ and $|\psi_2\rangle$ are either equal (up to a phase) or orthogonal. Then there is an exact procedure that returns $|0\rangle$ if the two states are orthogonal and $|1\rangle$ if they are equal. The procedure uses $O(1)$ applications of $\mathcal{P}_1$, $\mathcal{P}_2$ and their inverses and $O(\ell)$ elementary gates.*

Proof. We consider the quantum state valued function f from $\mathbb{Z}_2$ to $\mathbb{C}^{2\ell}$ such that $|f(0)\rangle = |\psi_1\rangle|\psi_2\rangle$ and $|f(1)\rangle = |\psi_2\rangle|\psi_1\rangle$. Note that $|f(0)\rangle$ and $|f(1)\rangle$ are either equal (if $|\psi_1\rangle = c|\psi_2\rangle$) or orthogonal. Also, if the two states are equal then the subgroup hidden by f is the entire group $\mathbb{Z}_2$ while in the other case it is the trivial subgroup. A procedure computing $|x\rangle|f(x)\rangle$ from $|x\rangle|0\rangle$ can be built using $\mathcal{P}_1$ and $\mathcal{P}_2$ and ℓ controlled swap of qubits. The proof can be concluded by an application of Theorem 1. □

We remark that as the HSP is in $\mathbb{Z}_2$, already the hidden subgroup algorithm of Brassard and Hoyer [6] implies the result. Also, by unfolding the hidden subgroup algorithm, the proof would give essentially the same circuit as a direct combination of the swap test and the

amplitude amplification would result in.

4.1.2 Testing membership

The swap test can be used to reduce Task (ii) of Theorem 2 to Task (i).

Corollary 3 *Let K be a subgroup of G and assume that we have a unitary procedure $\mathcal{S}_K$ for creating the uniform superposition $|K\rangle$ of the elements of K. Then we can test membership of $u \in G$ in K in time $\ell^{O(1)}$ by an exact quantum algorithm that uses $O(1)$ applications of the group oracles and $\mathcal{S}_K$ or the inverses of these.*

In the statement, the subgroup K is hidden behind the procedure $\mathcal{S}_K$. When we apply this corollary, $\mathcal{S}_K$ will be actually implemented by a procedure performing Task (i) of Theorem 2 for K in place of G. That procedure uses (generators for) K as input.

Proof. We apply Corollary 2 with $\mathcal{P}_1 = \mathcal{S}_K$ and $\mathcal{P}_2 = \mu_u \circ \mathcal{S}_K$ where $\mu_u(v) = uv$. □

4.1.3 Presentation of an abelian factor

The following corollary provides an algorithm to efficiently compute presentations and decomposition of abelian factors using the exact hidden subgroup algorithm.

Corollary 4 *Let K be a normal subgroup of G and assume that there is unitary procedure $\mathcal{S}_K$ that (on input $|0\rangle$) computes $|K\rangle$. Suppose further that we are given elements $u_1, \ldots, u_n \in G$ such that $u_i^{m^k} = 1$ for some integer $k \geq 0$; $[u_i, u_j] \in K$ $(i, j = 1, \ldots, n)$; and G is generated by $u_1, \ldots, u_n$ and K. Then there are exact quantum algorithms that, in time $(\ell \log m)^{O(1)}$, compute a presentation of G/K in terms of the generators $u_1K, \ldots, u_nK$ and an isomorphism $G/K \cong \mathbb{Z}_{m_1} \oplus \ldots \oplus \mathbb{Z}_{m_{n'}}$ with $m_i | m_{i-1}$ for $1 < i \leq n'$. The procedures apply $\mathcal{S}_K$, QFT_m, the group oracle(s) and the inverses of these. The aforementioned isomorphism will be given by listing n', the sequence $m_1, \ldots, m_{n'}$ and an n' by n matrix (α_{ij}) such that with $z_i = \prod_{j=1}^{n} u_j^{\alpha_{ij}}$ $(i, 1, \ldots, n')$, we have $z_i^{m_i} \in K$ but $\prod_{i=1}^{n'} z_i^{\beta_i} \in K$ with $0 \leq \beta_i < m_i$ implies that all $\beta_i = 0$.*

Proof. As the order of u is at most 2^ℓ, there exist an integer k between 0 and ℓ such that $u^{m^k} = 1$. (Let α_p be the multiplicity of the prime p in the order of u. Then $\alpha_p \leq \ell$ and if $k \geq \alpha_p$ for every prime factor p of m, we have $u^{m^k} = 1$.) We can find such a k using at most ℓ trials. Note that these trials can also be used to decide if the order of u is a divisor of a power of m.

Consider the function $f : \mathbb{Z}_{m^k}^n \to \mathbb{C}G$ defined as

$$f(x_1, \ldots, x_n) = \Big| \Big(\prod_{i=1}^{n} u_i^{x_i} \Big) K \Big\rangle.$$

The function f hides the subgroup H of $\mathbb{Z}_m^n$ such that L_H is the lattice of relations for a presentation of G/K in terms of generators u_iK $(i = 1, \ldots, n)$. We apply Theorem 1 to compute H. The result will actually be an Hermite normal form matrix whose columns are a basis for L_H. The Smith normal form of that matrix and the left multiplier give the stated isomorphism. □

We remark that Corollary 4 captures many tasks, such as finding orders of elements of G, finding generators for cyclic subgroups and solving the discrete logarithm problem for pairs of elements of G. It could also serve as an alternative method for testing membership in K.

4.2 Computing the uniform superposition of group elements

In this part, we show how to perform Task (i) of Theorem 2 provided that we have a polycyclic series $\{1\} = G_0 < G_1 < \ldots < G_h = G$ given by elements $g_1, \ldots, g_h$ such that G_i is a generated by $g_1, \ldots, g_i$ $(i = 1, \ldots, h)$. We also require that the order of each factor group G_i/G_{i-1} is a divisor of m. This will be ensured when we construct the polycyclic series. We use Watrous's method to build a "pyramid" of superpositions over the subgroups in the series. The key step is an exact version of Watrous's technique described in Section 3.2 of [19] that, from s copies of the uniform superposition $|G_i\rangle$ of elements of G_i computes $s-1$ copies of $|G_{i+1}\rangle$. Then creating $|G\rangle$ goes as follows. We start with $h+1$ copies of $|1\rangle$ and from these we compute $h+1-i$ copies of $|G_i\rangle$ $(i = 1, \ldots, h)$.

Below we describe the key step. As a tool, we use the following.

Lemma 1 *Let $z_1, \ldots, z_s$ be integers between 0 and $m-1$. Then there is a deterministic algorithm that in time $(s \log m)^{O(1)}$ finds integers $u_1, \ldots, u_{s-1}$ such that* $\gcd(u_1 z_1 + u_2 z_2 + \ldots + u_{s-1} z_{s-1} + z_s, m) = \gcd(z_1, \ldots, z_s, m)$.

Proof. There is an easy reduction to the case $s = 2$. Indeed, having a method for $s = 2$ we can find u_{s-1} such that $\gcd(u_{s-1} z_{s-1} + z_s, m) = \gcd(z_{s-1}, z_s, m)$ and then $\gcd(z_1, \ldots, z_{s-2}, u_{s-1} z_{s-1} + z_s, m) = \gcd(z_1, \ldots, z_{s-1}, z_s, m)$. We proceed with finding an appropriate coefficient u_{s-2}, and so on.

To solve the case $s = 2$, given z_1, z_2 and m, we need to find u such that $\gcd(uz_1 + z_2, m) = \gcd(z_1, z_2, m)$. Dividing by $\gcd(z_1, z_2, m)$, we obtain the case when $\gcd(z_1, z_2, m) = 1$ and when we are looking for u such that $\gcd(uz_1 + z_2, m) = 1$. For each prime divisor p of m we say that u is "bad" modulo p if $p | uz_1 + z_2$, otherwise p is "good" modulo p. Obviously, goodness and badness depend only on the residue class of u modulo p and for every p there are $p-1$ "good" residue classes. Let S be a positive integer to be determined later and let P be the set of the prime divisors of m less than S. These primes can be listed and their product m' can be computed in time $(S \log m)^{O(1)}$. For each prime in P pick a "good" residue class u_p modulo p and by Chinese remaindering compute an integer $0 \leq u_0 < m'$ such that $u_0 \equiv u_p$ modulo p for every $p \in P$. Then for every integer t, $u_t = u_0 + tm'$ is good modulo every prime in P. We take the sequence $u_0, \ldots, u_{S-1}$ and compute $\gcd(u_t z_1 + z_2, m)$ for $t = 0, \ldots, S-1$. Clearly, for each prime $p > S$, the sequence does not contain two members that are from the same residue class modulo p. Therefore, at most one of the u_ts is bad modulo p. Thus, if $S > \log_2 m$ then there is at least one t such that u_t is good modulo every prime divisor of m. We take that u_t (reduced modulo m). □

Let N be a subgroup of G and let $u \in G$ such that $u^{-1} N u = N$ and $u^m \in N$. Let K be the subgroup generated by u and N. Then $N \lhd K$. Assume that we have s copies of the state $|N\rangle = \frac{1}{\sqrt{|N|}} \sum_{v \in N} |v\rangle$. From these, we shall make $s-1$ copies of the state $|K\rangle = \frac{1}{\sqrt{|K|}} \sum_{w \in K} |w\rangle$. We use Watrous's method with two modifications. The first thereof is that we take the multiple m instead of the order of u modulo N so that we do not need to use any Fourier transform other than QFT_m. The second modification is that, as we want an exact procedure, we have to be prepared to handle some "degenerate" cases that occur with small probability. (Watrous discarded these and used repetition. However, the possibility of a variant that also works in the degenerate cases is mentioned in [19].)

Notice that $|K\rangle = \frac{1}{\sqrt{m}} \sum_{x \in \mathbb{Z}_m} |u^x N\rangle$. We apply the Fourier sampling described in Subsec-

tion 2.3 for the function $f : \mathbb{Z}_m \to \mathbb{C}G$ given as $f(x) = |u^x N\rangle$. This time we are not interested which subgroup is hidden by f, our objectives are the states $|\gamma_y\rangle$ in the second register of $|\Psi\rangle$ in (2) and in (3). In this special case,

$$|\gamma_y\rangle = \frac{1}{\sqrt{m}} \sum_{x \in \mathbb{Z}_m} \omega^{xy} |u^x N\rangle.$$

(Here we go back to (1) and do not use the decomposition of $|\gamma_y\rangle$ by the cosets of the hidden subgroup.) Another minor difference from Subsection 2.3 is that we compute f in a register initially $|N\rangle$ rather than $|0\rangle$.

By applying Fourier sampling on the available s copies of $|N\rangle$, we obtain

$$(m)^{-s/2} \sum_{y_1,\ldots,y_s=0}^{m-1} |y_1\rangle \ldots |y_s\rangle |\gamma_{y_1}\rangle \ldots |\gamma_{y_s}\rangle.$$

We consider the terms of this sum. For each s-tuple $(y_1, \ldots, y_s)$ we will change the first $s-1$ states to $|\gamma_0\rangle = |K\rangle$ using Watrous's trick. The key fact behind is that $|\gamma_y\rangle$ is an eigenstate with eigenvalue ω^{-y} for the action of u by multiplication. Based on this, it is easy to show that if we have a pair $|\gamma_y\rangle|\gamma_z\rangle$ of such states then multiplying the content of the first part by the tth power of the content of the second part, the effect can be interpreted as the first part $|\gamma_y\rangle$ remains unchanged while the second part becomes $|\gamma_{z-ty}\rangle$, where $z - ty$ is understood modulo m. We use this first to arrange that the index y_s of the last state γ_{y_s} becomes a generator for the subgroup L of $\mathbb{Z}_m$ generated by $y_1, \ldots, y_s$. To this end, using the method of Lemma 1, we find integers $u_1, \ldots, u_{s-1}$ such that $\gcd(u_1 y_1 + \ldots + u_{s-1} y_{s-1} + y_s, m) = \gcd(y_1, \ldots, y_s, m)$. Then $u_1 y_1 + \ldots + u_{s-1} y_{s-1} + y_s$ modulo m is a generator for L. We multiply (the content of) $|\gamma_{y_1}\rangle$ by the $-u_1$th power of $|\gamma_{y_s}\rangle$, then $|\gamma_{y_2}\rangle$ the $-u_2$th power of by an appropriate power of (the new) $|\gamma_{y_s}\rangle$, and so on. Eventually y_s becomes the generator $u_1 y_1 + \ldots + u_{s-1} y_{s-1} + y_s$ modulo m. Then $y_i \equiv t_i y_s$ modulo m for some integer t_i $(i = 1, \ldots, s-1)$, and multiplying $|\gamma_{y_s}\rangle$ with the t_ith power of $|\gamma_{y_i}\rangle$ changes $|\gamma_{y_i}\rangle$ to $|\gamma_{y_i - t_i y_s}\rangle = |\gamma_0\rangle = |K\rangle$.

4.3 Constructing a polycyclic series

We start with testing if the orders of the generators are divisors of some power of m. (The proof of Corollary 4 includes a straightforward classical method doing this.) If one of the tests fails, then no further computation is needed: we can return that the order of G does not satisfy the required property. When a new element occurs, we can perform this test and exit in case of failure. We build a polycyclic series from the bottom.

Assume that $N \lhd G$ and we already have a polycyclic series of N with factors of order dividing m. Initially, N is the trivial subgroup. By the previous subsection, we have an efficient method for Task (i) of Theorem 2 with N in place of G and one can use Corollary 3 (or Corollary 4) to test membership in N. Our first goal is to find a normal subgroup $K \lhd G$ such that K/N is abelian. To this end, we use a black box method from Luks's paper [23]. Let $y_1, \ldots, y_{r'}$ be those of the given generators for G that are not in N. We may assume that $r' \geq 1$ as otherwise $G = N$. Put $x = y_1$. We collect a list L of conjugates of x until we find two elements $u, v \in L$ such that $[u, v] \notin N$ or the subgroup K generated by L and N stabilizes. Initially L contains only x. When a new conjugate v of x is added to the list then

we check if $[u, v] \in N$ for every u already in L. If an u is found such that $[u, v] \notin N$ then we replace x with $[u, v]$. We test if the order of the new x is a divisor of some power of m and stop if not. Otherwise, we restart collecting conjugates of x.

Stabilization can be decided as follows. Notice that K/N is abelian and L directly extends the polycyclic series of N to K. The series can also be refined to have factors of order dividing m in an obvious way. Based on this, there is an efficient method for Task (i) of Theorem 2 for K as well, whence we can test membership in K like in N. Using that, for every u in the list L we test if $u^{y_i} \in K$ for every y_i. If this is the case, then $K \lhd G$. Otherwise, we can add a new conjugate z^{y_i} of x to L. When K stabilizes, we replace N with K and repeat the procedure outlined above. As $|L| \leq \log_2 |K/N| \leq \ell$, either K stabilizes or two conjugates of x is found with commutator not in N in at most $\log_2 G \leq \ell$ rounds.

If G/N is solvable and $N < G$ then one eventually finds a normal subgroup K of G properly containing N because if $x \in G^{(i)}$ for some i and if u, v are conjugates of x then $[u, v] \in [G^{(i)}, G^{(i)}] = G^{(i+1)}$. Therefore, to detect non-solvability, we keep track how many times the element x has been updated. If it has happened more than ℓ times, then we can stop and conclude that G is not solvable.

4.4 The order, testing membership and abelian factors

The order of G is the product of the orders of the factors G_{i+1}/G_i of a polycyclic series $\{1\} = G_0 < G_1 \ldots < G_h = G$. The order of a factor can be computed by the algorithm of Corollary 4. Testing membership in a subgroup $N \leq G$ can be done by building a polycyclic series of N and then using Corollary 3. (For testing membership of x in G itself, we replace N with G and G with the group generated by x and G.) Similarly, decomposing an abelian factor G/N can be done by constructing first a polycyclic series of N and then applying Corollary 4.

4.5 The derived series

We first show how to compute the commutator subgroup G'. The next to last element in a polycyclic series is a normal subgroup $N \lhd G$ such that G/N is cyclic. The data structure for the polycyclic series algorithm includes an element g such that g and N generate G. By recursion, we start with computing the commutator subgroup N'. As N' is a characteristic subgroup of N, we have $N' \lhd G$. We can also find a polycyclic series for N' which enables us computing $|N'\rangle$. We take the iterated commutators of the generators for N with g until the subgroup generated by N and these commutators stabilizes. This subgroup will be G'. To check stabilization, we use Corollary 3 to test membership in the intermediate subgroups.

By recursion, we compute the derived series of G' which extends to that of G in the obvious way.

Acknowledgements

The authors are grateful to Lajos Rónyai, to Igor Shparlinski and to an anonymous referee for their helpful comments and suggestions. The research of the second author was supported by the Hungarian Ministry of Innovation and Technology NRDI Office within the framework of the Artificial Intelligence National Laboratory Program.

References

1. D. Deutsch and R. Jozsa (1992), *Rapid solution of problems by quantum computation*, Proc. Math. Phys. Eng. Sci., Vol. 439, pp. 553 – 558.
2. E. Bernstein and U. Vazirani (1997), *Quantum Complexity Theory*, SIAM J. Comput., Vol. 26, pp. 1411–1473.
3. D. Simon (1997), *On the Power of Quantum Computation*, SIAM J. Comput., Vol. 26, pp. 1474–1483.
4. A. Kitaev (1996), *Quantum measurements and the Abelian Stabilizer Problem*, Electron. Colloquium Comput. Complex., Vol. 3.
5. M. Ettinger, P. Høyer and E. Knill (2004), *The quantum query complexity of the hidden subgroup problem is polynomial*, Inf. Process. Lett., Vol. 91, pp. 43–48.
6. G. Brassard and P. Høyer (1997), *An exact quantum polynomial-time algorithm for Simon's problem*, In *ISTCS 97*. pp. 12–23.
7. L. Grover (1997), *Quantum mechanics helps in searching for a needle in a haystack*, Phys. Rev. Lett., Vol. 79, no. 2, pp. 325–328, `arXiv:quant-ph/9706033`.
8. T. Mihara and S. Sung (2003), *Deterministic polynomial-time quantum algorithms for Simons problem*, Comput. Complex., Vol. 12, pp. 162–175.
9. X. Bonnetain (2021), *Tight Bounds for Simon's Algorithm*, In *LATINCRYPT 2021*. pp. 2–23.
10. M. Mosca and C. Zalka (2003), *Exact quantum Fourier transforms and discrete logarithm algorithms*, Int. J. Quantum Inf., Vol. 02, pp. 91–100.
11. R. Cleve (1994), *A note on computing Fourier transforms by quantum programs*, Manuscript, University of Calgary, http://pages.cpsc.ucalgary.ca/∼cleve/papers.html.
12. D. Coppersmith (1994), *An approximate Fourier transform useful in quantum factoring", IBM Research Report RC19642*, Also at https://arXiv.org/abs/quant-ph/0201067.
13. H. Nishimura and M. Ozawa (2002), *Computational complexity of uniform quantum circuit families and quantum Turing machines*, Theor. Comput. Sci., Vol. 276, pp. 147–181.
14. H. Nishimura and M. Ozawa (2005), *Uniformity of quantum circuit families for error-free algorithms*, Theor. Comput. Sci., Vol. 332, pp. 487–496.
15. H. Nishimura and M. Ozawa (2009), *Perfect computational equivalence between quantum Turing machines and finitely generated uniform quantum circuit families*, Quantum Inf. Process., Vol. 8, pp. 13–24.
16. M. Mosca (2002), *On the quantum derandomization of algorithms*, Presentation at the MSRI workshop on Quantum Information Processing, https://www.msri.org/workshops/204/schedules/1233.
17. J. Li, X. Peng, J. Du and D. Suter (2012), *An Efficient Exact Quantum Algorithm for the Integer Square-free Decomposition Problem*, Sci. Rep., Vol. 2, pp. 1–5.
18. H. Buhrman, R. Cleve, J. Watrous and R. de Wolf (2001), *Quantum fingerprinting.*, Phys. Rev. Lett., Vol. 87/16, p. 167902.
19. J. Watrous (2001), *Quantum algorithms for solvable groups.*, In *STOC 2001*. pp. 60–67.
20. P. Koiran, V. Nesme and N. Portier (2007), *The quantum query complexity of the abelian hidden subgroup problem*, Theor. Comput. Sci., Vol. 380, pp. 115–126.
21. A. Nayak (2021), *Deterministic Algorithms for the Hidden Subgroup Problem*, Tech. Rep. 2104.1436 [cs.DS], arXiv.
22. L. Babai, G. Cooperman, L. Finkelstein, E. Luks and A. Seress (1995), *Fast Monte Carlo algorithms for permutation groups*, J. Comput. Syst. Sci., Vol. 50, no. 2, pp. 296–308.
23. E. Luks (1992), *Computing in solvable matrix groups*, FOCS 1992, pp. 111–120.
24. Y. Wang (1959), *On the least primitive root of a prime*, Acta Math. Sinica, Vol. 9, pp. 432–441.
25. E. Bach (1997), *Comments on search procedures for primitive roots*, Math. Comput., Vol. 66, pp. 1719–1727.
26. V. Shoup (1992), *Searching for primitive roots in finite fields*, Math. Comput., Vol. 58, pp. 369–380.
27. I. Shparlinski (1992), *On primitive element in finite fields and on elliptic curves*, Math. USSR-Sb., Vol. 71, pp. 41–50.

28. G. I. Perel'muter and I. E. Shparlinski (1990), *The distribution of primitive roots in finite fields (in Russian)*, Russ. Math. Surv., Vol. 45, pp. 185–186.
29. I. E. Shparlinski (2018), *On Constructing Primitive Roots in Finite Fields With Advice*, IEEE T. Inform. Theory, Vol. 64, pp. 7132–7136.
30. D. Robinson (1995), *A Course in the Theory of Groups* (Springer), 2nd ed.
31. L. Babai and E. Szemerédi (1984), *On the Complexity of Matrix Group Problems I*, In *FOCS 1984*. pp. 229–240.
32. G. Ivanyos, A. Joux and M. Santha (2019), *Discrete logarithm and Diffie-Hellman problems in identity black-box groups*, Tech. Rep. 1911.01662 [quant-ph], arXiv.
33. R. Kannan and A. Bachem (1979), *Polynomial Algorithms for Computing the Smith and Hermite Normal Forms of an Integer Matrix*, SIAM J. Comput., Vol. 8, pp. 499–507.