Modifying Shor's algorithm to compute short discrete logarithms

Martin Ekerå*

December 7, 2016

Abstract

We revisit Shor's algorithm for computing discrete logarithms in $\mathbb{F}_p^*$ on a quantum computer and modify it to compute logarithms d in groups $\langle g \rangle$ of prime order q in the special case where $d \lll q$. As a stepping stone to performing this modification, we first introduce a modified algorithm for computing logarithms on the general interval $0 < d < q$ for comparison.

We demonstrate conservative lower bounds on the success probability of our algorithms in both the general and the special case. In both cases, our algorithms initially set the index registers to a uniform superposition of all states, compared to $p-1$ states in Shor's original algorithm.

In the special case where $d \lll q$, our algorithm uses $3\,\lceil \log_2 d \rceil$ qubits for the two index registers and computes two QFTs of size $2^{\lceil \log_2 d \rceil}$ and $2^{2\,\lceil \log_2 d \rceil}$, compared to $2\,\lfloor \log_2 q \rfloor$ qubits for the index registers and two QFTs both of size $2^{\lfloor \log_2 q \rfloor}$ in the general case.

A quantum circuit for computing $\lfloor a - bd \rceil\, g$ is furthermore required, where $0 \leq a < 2^{2\,\lceil \log_2 d \rceil}$ and $0 \leq b < 2^{\lceil \log_2 d \rceil}$ in the special case, compared to $0 \leq a, b < 2^{\lfloor \log_2 q \rfloor}$ in the general case.

This implies that the complexity of computing discrete logarithms on a quantum computer can be made to depend not only on the choice of group, and on its order q, but also on the logarithm d.

In the special case where $d \lll q$, our algorithm does not require q to be prime. It may hence be generalized to finite abelian groups.

1 Introduction

In a groundbreaking paper [10] from 1994, subsequently extended and revised in a later publication [11], Shor introduced polynomial time quantum computer algorithms for factoring integers over $\mathbb{Z}$ and for computing discrete logarithms in the multiplicative group $\mathbb{F}_p^*$ of the finite field $\mathbb{F}_p$.

Although Shor's algorithm for computing discrete logarithms was originally described for $\mathbb{F}_p^*$, it may be generalized to any finite abelian group, provided the group operation may be implemented efficiently using quantum circuits. This generalization was first described by Boneh and Lipton [1]. Shor's algorithms may furthermore be generalized by perceiving them as algorithms for solving special instances of the finite abelian hidden subgroup problem [4].

*KTH Royal Institute of Technology, SE-100 44 Stockholm, Sweden and Swedish NCSA, Swedish Armed Forces, SE-107 85 Stockholm, Sweden. Use ekera@kth.se to e-mail author.

Virtually all asymmetric cryptographic schemes that are widely deployed today rely on the computational intractability of either the integer factoring problem or the discrete logarithm problem in some abelian group that meets the above criteria. Consequently the work of Shor has the potential to greatly impact the field of asymmetric cryptography.

In this paper, we revisit Shor's algorithm for solving the discrete logarithm problem. In section 1.1 below, we introduce the discrete logarithm problem, discuss the choice of groups and introduce a special case of the discrete logarithm problem that we study in this paper along with a rationale for why it merits study. In section 1.2 we proceed to describe our contributions and in section 1.3 we give an overview of the remainder of this paper.

1.1 The discrete logarithm problem

To formally introduce the discrete logarithm problem, let $\mathbb{G}$ under $\odot$ be a group generated by g, and let

$$x = [d]\, g = \underbrace{g \odot g \odot \cdots \odot g \odot g}_{d \text{ times}}.$$

Given x, a generator g and a description of $\mathbb{G}$ and $\odot$ the discrete logarithm problem is to compute $d = \log_g x$.

Different groups are used to instantiate cryptographic schemes that rely on the computational intractability of the discrete logarithm problem. Common choices include subgroups of $\mathbb{F}_p^*$ and elliptic curve groups $E(\mathbb{K})$ on Weierstrass form, Edwards form, twisted Edwards form or Montgomery form, where $\mathbb{K}$ is some field such as $\mathbb{F}_p$ or $\mathbb{F}_{2^n}$.

The bracket notation that we have introduced above is commonly used in the literature to denote repeated application of the group operation regardless of whether the group is written multiplicatively or additively.

In $\mathbb{F}_p^*$ we have $[d]\, g = g^d \bmod p$. In the case of elliptic curve groups $[d]\, g$ denotes repeated addition of the point g to itself d times.

1.1.1 Groups of prime order q

As was first described by Pohlig and Hellman [7], the discrete logarithm problem in a finite abelian group may be decomposed into multiple discrete logarithm problems in subgroups of prime order or prime power order to the parent group, assuming the factorization of its order is known. The complexities of solving these problems depend in general on the sizes of the subgroups.

Once the discrete logarithm problems in these subgroups have been solved, the results may be combined using the Chinese remainder theorem to yield a solution to the discrete logarithm problem in the parent group.

Consequently, prime order groups are used more or less exclusively when instantiating cryptographic schemes that rely on the conjectured computational intractability of the discrete logarithm problem. We therefore primarily consider prime order groups in this paper.

1.1.2 Subgroups of prime order to $\mathbb{F}_p^*$

To compute discrete logarithms in prime order q subgroups of $\mathbb{F}_p^*$ for large prime p that are not on special form, the best classical algorithm is the general number

field sieve (GNFS). The time complexity of the GNFS when adapted to compute discrete logarithms is $\mathcal{O}(\exp[(\frac{64}{9})^{\frac{1}{3}} (\ln p)^{\frac{1}{3}} (\ln \ln p)^{\frac{2}{3}}])$ as shown in Schirokauer's [9] improvements of Gordon's [2] original adaptation.

Another option in the classical setting is to use cycle finding algorithms such as Pollard-ρ, the time complexity of which is $\mathcal{O}\left(\sqrt{q}\right)$, or Pollard-$\lambda$, the time complexity of which is $\mathcal{O}(\sqrt{d})$. Pollard's algorithms [8] are generic in the sense that they solve the discrete logarithm problem in any prime order group.

In instantiations of cryptographic schemes that rely on the computational intractability of the discrete logarithm problem, the prime p must hence be selected sufficiently large to resist attacks based on the GNFS, whilst d and q must only be selected sufficiently large for attacks based on the cycle finding algorithms to have higher time complexity than attacks based on the GNFS.

Hence, it is possible to select $q \lll p$. Such group are often referred to as Schnorr groups. In instantiations, it is typically either the case that $q \lll p$, or that $q \sim p$ where often $p = 2q + 1$ is a so-called safe prime. With respect to known classical attacks, these options offer a comparable level of security.

Since $d < q < p$ it is possible to always select $d \sim q$. However, in the case where $q \sim p$ it is advantageous to select d so that $d \lll q \sim p$ as opposed to $d \sim q \sim p$, since d being small speeds up the modular arithmetic. Again, with respect to classical attacks, these options offer a comparable level of security. For a more in-depth analysis, see the work [6] of van Oorschot and Wiener.

1.1.3 On our interest in the special case $d \lll q$

If Shor's original algorithm is used to compute discrete logarithms in subgroups of prime order q to $\mathbb{F}_p^*$ on a quantum computer, the complexity of the algorithm depends on p. This implies that the case $q \lll p$ is as complex as the case $q \sim p$, which is not in analogy with the classical algorithms.

If the algorithm is modified to instead compute discrete logarithms in groups of prime order q, as in section 4, its complexity can be made to depend on p and q so that the case $q \lll p$ becomes less complex than the case $q \sim p$.

However, the complexity still does not depend on d. This implies that the case $d \lll q \sim p$ is more complex than the case $d \sim q \lll p$, which again is not in analogy with the classical algorithms. This observation, coupled with the observation that using small d is advantageous in practical implementations for performance reasons, served as one of our motivations for modifying Shor's algorithm to compute discrete logarithms in the special case where $d \lll q$.

Another motivation was our interest in understanding how the complexity of computing discrete logarithms in elliptic curve groups relates to the complexity of computing discrete logarithms in subgroups of $\mathbb{F}_p^*$.

1.1.4 Discrete logarithms on short intervals

In cases where d is not short in the sense that $d \lll q$, but is known to be constrained to some short interval $d_1 \leq d < d_2$, it must be the case that

$$x = [d]\, g = [d_1]\, g \odot [d - d_1]\, g \;\Rightarrow\; x' = x \odot [-d_1]g = [d - d_1]\, g = [d']\, g.$$

To compute d in such cases, it is hence possible to first compute x', to then compute the discrete logarithm d', that is on the interval $0 \leq d' < (d_2 - d_1)$ and hence short, from x' and g, and to finally compute $d = d_1 + d'$.

This implies that any algorithm for efficiently computing d when $d \lll q$ may be used to also efficiently compute d when d is on a short interval.

1.2 Our contributions

In this paper, we revisit Shor's algorithm for computing discrete logarithms in $\mathbb{F}_p^*$ on a quantum computer and modify it to compute logarithms d in groups of prime order q in the special case where $d \lll q$. As a stepping stone to performing this modification, we first introduce a modified algorithm for computing logarithms on the general interval $0 < d < q$ for comparison.

We provide a complete analysis of our algorithms, both in the special case where $d \lll q$ and in the general case. In particular, we demonstrate conservative lower bounds on their success probabilities using simple proof techniques.

Shor's original algorithm for computing discrete logarithms uses two index registers that are initialized to a superposition of $p - 1$ states. This translates into a superposition of q states in the case where the group order is a prime q.

Our algorithms instead initialize these registers to a uniform superposition of all states. This may be advantageous from an implementation perspective since a uniform superposition of all states may be efficiently induced, for example by applying the Hadamard transform to the zero state.

In the special case where $d \lll q$, our algorithm uses $3 \lceil \log_2 d \rceil$ qubits for the two index registers and requires the computation of two QFTs of size $2^{\lceil \log_2 d \rceil}$ and $2^{2 \lceil \log_2 d \rceil}$ respectively, compared to $2 \lfloor \log_2 q \rfloor$ qubits for the index registers and two QFTs both of size $2^{\lfloor \log_2 q \rfloor}$ in the general case.

A quantum circuit capable of computing $[e]\, g = [a]\, g \odot [-b]\, x$ is furthermore required, where $x = [d]\, g$, and where $0 \leq a < 2^{2 \lceil \log_2 d \rceil}$ and $0 \leq b < 2^{\lceil \log_2 d \rceil}$ in the special case, compared to $0 \leq a, b < 2^{\lfloor \log_2 q \rfloor}$ in the general case.

These results imply that the complexity of computing discrete logarithms in prime order groups on a quantum computer can be made to depend not only on the choice of group, and on its order q, but also on the logarithm d.

In the special case where $d \lll q$, our algorithm does not require q to be prime. Hence the algorithm generalizes to finite abelian groups.

1.3 Overview of this paper

In section 2 below we introduce some notation that is used throughout this paper and in section 3 we provide a brief introduction to quantum computing.

In section 4 we proceed to describe our algorithm for computing discrete logarithms in groups of prime order q in the general case where $0 < d < q$, and in section 5 we describe our modified algorithm for the special case $d \lll q$. We conclude the paper and summarize the results in section 6.

2 Notation

In this section, we introduce some notation for norms, rounding operations and modular reduction operations that are used throughout this paper.

- $\lfloor u \rceil$ denotes u rounded to the closest integer.
- $\lfloor u \rfloor$ denotes u rounded to the closest integer less than or equal to u.

- $\lceil u \rceil$ denotes u rounded to the closest integer greater than or equal to u.
- $u \bmod n$ denotes u reduced modulo n and constrained to the interval

$$0 \leq u \bmod n < n.$$

- $\{u\}_n$ denotes u reduced modulo n and constrained to the interval

$$-n/2 \leq \{u\}_n < n/2.$$

- $u \bmod 1 = u - \lfloor u \rfloor$ denotes the fractional part of u.
- $|\, a + ib \,| = \sqrt{a^2 + b^2}$ where $a, b \in \mathbb{R}$ denotes the Euclidean norm of $a + ib$ which is equivalent to the absolute value of a when b is zero.
- If $\vec{u} = (u_0, \ldots, u_{n-1}) \in \mathbb{R}^n$ is a vector then

$$|\, \vec{u} \,| = \sqrt{u_0^2 + \ldots + u_{n-1}^2}$$

  denotes the Euclidean norm of $\vec{u}$.

3 Quantum computing

In this section, we provide a brief introduction to quantum computing and introduce some notation and terminology that is used throughout this paper. The contents of this section is to some extent a layman's description of quantum computing, in that it may leave out or overly simplify important details.

There is much more to be said on the topic of quantum computing. However, such elaborations are beyond the scope of this paper. For more information, the reader is instead referred to [3]. The extended paper [11] by Shor also contains a very good introduction and many references to the literature.

3.1 Quantum systems

In a classical electronic computer, a register that consists of n bits may assume any one of 2^n distinct states j for $0 \leq j < 2^\ell$. The current state of the register may be observed at any time by reading the register.

In a quantum computer, information is represented using qubits; not bits. A register of n qubits may be in a superposition of 2^n distinct states. Each state is denoted $|\, j \,\rangle$ for $0 \leq j < 2^n$ and a superposition of states, often referred to as a quantum system, is written as a sum

$$|\, \Psi \,\rangle = \sum_{j=0}^{2^n-1} c_j \,|\, j \,\rangle \quad \text{where} \quad c_j \in \mathbb{C} \quad \text{and} \quad \sum_{j=0}^{2^n-1} |\, c_j \,|^2 = 1$$

that we shall refer to as the system function.

Each complex amplitude c_j may be written on the form $c_j = a_j e^{i\theta_j}$, where $a_j \in \mathbb{R}$ is a non-negative real amplitude and $0 \leq \theta_j < 2\pi$ is a phase, so the system function may equivalently be written on the form

$$|\, \Psi \,\rangle = \sum_{j=0}^{2^n-1} a_j e^{i\theta_j} \,|\, j \,\rangle \qquad \text{where} \qquad \sum_{j=0}^{2^n-1} a_j^2 = 1.$$

3.2 Measurements

Similar to reading a register in a classical computer, the qubits in a register may be observed by measuring the quantum system.

The result of such a measurement is to collapse the quantum system, and hence the system function, to a distinct state. The probability of the system function $|\,\Psi\,\rangle$ collapsing to $|\,j\,\rangle$ is $|\,c_j\,|^2 = a_j^2$.

3.3 Quantum circuits

It is possible to operate on the qubits that make up a quantum system using quantum circuits. Such circuits are not entirely dissimilar from the electrical circuits used to perform operations on bit registers in classical computers.

3.3.1 Quantum operators

For a quantum circuit to be physically permissible it must be reversible and it must preserve the normalization of the system function.

If the system function $|\,\Psi\,\rangle$ that describes the state of the quantum system is perceived as a column vector of 2^n complex amplitudes c_j then the set of all permissible quantum circuits is described by the set of all unitary $2^n \times 2^n$ matrices with elements in $\mathbb{C}$.

Executing the circuit described by U with respect to a quantum system that is described by the system function $|\,\Psi\,\rangle$ transforms the quantum system into a new quantum system that is described by the system function $U\,|\,\Psi\,\rangle$. Therefore the unitary matrix U is said to be a quantum operator.

3.3.2 Constructing quantum circuits

Any two quantum operators may be arbitrarily composed under multiplication since the product of any two unitary matrices is itself a unitary matrix. It may furthermore be shown that large quantum operators may be built from smaller operators by taking matrix tensor products.

A quantum circuit for a large register may hence be constructed by taking products of and tensoring small quantum operators that act as building blocks. A set of quantum operators is said to be universal if any circuit, up to some degree of precision, may be constructed using only operators from the set.

The quantum algorithms in this paper are expressed in purely mathematical terms. To be executed on a physical quantum computer, these algorithms must first be compiled into circuits that can be realized physically by the computer.

3.4 Quantum-mechanical entanglement

Two quantum systems are said to be separable if and only if the system function may be written as a tensor product $|\,\Psi\,\rangle = |\,\Psi_1\,\rangle \otimes |\,\Psi_2\,\rangle$ where $|\,\Psi_1\,\rangle$ involves only states from a first register and $|\,\Psi_2\,\rangle$ involves only states from a second register. Otherwise, the two systems are said to be entangled.

Entangled systems are written on the form $|\,\Psi\,\rangle = |\,\Psi_1, \Psi_2\,\rangle$. Separable systems may be written on the form $|\,\Psi\,\rangle = |\,\Psi_1\,\rangle \otimes |\,\Psi_2\,\rangle = |\,\Psi_1\,\rangle\,|\,\Psi_2\,\rangle$.

If $|\,\Psi_1\,\rangle$ and $|\,\Psi_2\,\rangle$ are two entangled systems, the probability distributions that the systems represent are dependent, so that if $|\,\Psi_1\,\rangle$ is observed and it

collapses to some distinct state $|\, j \,\rangle$ then $|\, \Psi_2 \,\rangle$ collapses accordingly to a system that represents the original probability distribution conditioned on $|\, \Psi_1 \,\rangle = |\, j \,\rangle$.

The ability of quantum systems to be in superpositions of many different quantum states simultaneously, of quantum circuits to operate on all states in a quantum system simultaneously, and of quantum systems to be entangled, seemingly enables quantum computers to perform certain computations much faster than is theoretically possible with classical computers.

3.5 Probability amplification

Given a quantum system in some known initial state, the purpose of a quantum circuit is to amplify the amplitudes of a set of desired states, and to suppress the amplitudes of all other states, so that when the system is observed, the probability is large that it will collapse to a desired state. This concept is commonly referred to as probability amplification.

3.5.1 The quantum Fourier transform

In Shor's algorithms, that are the focus of this paper, the discrete quantum Fourier transform (QFT) is used to to achieve probability amplification.

The QFT maps each state in an n qubit register to

$$|\, j \,\rangle \quad \xrightarrow{\text{QFT}} \quad \frac{1}{\sqrt{2^n}} \sum_{k\,=\,0}^{2^n-1} e^{2\pi i\, jk/2^n} \,|\, k \,\rangle$$

so the QFT maps the system function

$$|\, \Psi \,\rangle = \sum_{j\,=\,0}^{2^n-1} c_j \,|\, j \,\rangle \quad \xrightarrow{\text{QFT}} \quad \frac{1}{\sqrt{2^n}} \sum_{j\,=\,0}^{2^n-1} \sum_{k\,=\,0}^{2^n-1} c_j \, e^{2\pi i\, jk/2^n} \,|\, k \,\rangle \,.$$

It is easy to see that the QFT is a unitary operator and hence permissible. It is furthermore efficient from an implementation perspective, see Shor's original paper [11] for more information.

3.5.2 Constructive interference

If the above system is observed, the probability of it collapsing to k is

$$\frac{1}{2^n} \cdot \left| \sum_{j\,=\,0}^{2^n-1} c_j \, e^{2\pi i\, jk/2^n} \right|^2 .$$

Perceive the terms in the sum as vectors in $\mathbb{C}$. If the vectors are of approximately the same norm and point in approximately the same direction, then the norm of their sum is likely to be great giving rise to a large probability. For k such that this is indeed the case, constructive interference is said to arise.

The claim below summarizes the notion of constructive interference that we use in this paper. Note that in our case, all vectors in the sum are unit vectors.

Claim 1. *Let θ_j for $0 \leq j < N$ be phase angles such that $|\,\theta_j\,| \leq \frac{\pi}{4}$. Then*

$$\left| \sum_{j=0}^{N-1} e^{i\theta_j} \right|^2 \geq \frac{N^2}{2}.$$

Proof.

$$\left| \sum_{j=0}^{N-1} \mathrm{e}^{i\theta_j} \right|^2 = \left| \sum_{j=0}^{N-1} (\cos\theta_j + i \sin\theta_j) \right|^2 \geq \left| \sum_{j=0}^{N-1} \cos\theta_j \right|^2 \geq \frac{N^2}{2}$$

since for j on the interval $0 \leq j < N$ we have $|\,\theta_j\,| \leq \frac{\pi}{4}$ which implies

$$\frac{1}{\sqrt{2}} \leq \cos\theta_j \leq 1$$

and so the claim follows. ■

4 Computing d in the general case

In this section we explain in detail how Shor's original algorithm for computing discrete logarithms in $\mathbb{F}_p^*$ may be adapted to computing discrete logarithms in a group $\mathbb{G}$ of known prime order q provided that the group operation $\odot$, and in particular the exponentiations, may be implemented efficiently using quantum circuits. Shor describes [11] how this may be accomplished for $\mathbb{F}_p^*$.

Upon input of a generator g of $\mathbb{G}$ and an element $x = [d]\,g$ where $0 < d < q$ the algorithm computes the discrete logarithm d. The algorithm consists of two parts; a quantum algorithm and a classical algorithm.

- The quantum algorithm is described in section 4.1.

  It accepts as input a generator g and an element $x = [d]\,g$ and produces as output a pair (j, k) and an element $[e]\,g$ that is ignored.

  In section 4.1.2 we identify a subset of pairs that we call good pairs and analyze the probability of a good pair being returned.

- The classical algorithm is described in section 4.2.

  It accepts as input the pair (j, k) output by the quantum algorithm and yields the discrete logarithm d if the pair (j, k) is good and if it fulfills certain additional criteria that are elaborated on in section 4.2.

4.1 The quantum algorithm for computing (j, k)

In this section we provide a detailed description of the quantum algorithm that upon input of g and $x = [d]\,g$ outputs a pair (j, k).

1. Let ℓ be the largest positive integer such that $2^\ell < q$ and let

   $$|\,\Psi\,\rangle = \frac{1}{2^\ell} \cdot \sum_{a=0}^{2^\ell-1} \sum_{b=0}^{2^\ell-1} |\,a\,\rangle\,|\,b\,\rangle\,|\,0\,\rangle$$

   where the first two registers are of length ℓ qubits.

2. Compute $[a]\,g \odot [-b]\,x$ and store the result in the third register

$$|\,\Psi\,\rangle = \frac{1}{2^\ell} \cdot \sum_{a=0}^{2^\ell-1} \sum_{b=0}^{2^\ell-1} |\,a, b, [a]\,g \odot [-b]\,x\,\rangle$$
$$= \frac{1}{2^\ell} \cdot \sum_{a=0}^{2^\ell-1} \sum_{b=0}^{2^\ell-1} |\,a, b, [a\ -\ bd]\,g\,\rangle\,.$$

3. Compute two QFTs of size 2^ℓ of the first two registers to obtain

$$|\,\Psi\,\rangle = \frac{1}{2^\ell} \cdot \sum_{a=0}^{2^\ell-1} \sum_{b=0}^{2^\ell-1} |\,a, b, [a\ -\ bd]\,g\,\rangle \quad \xrightarrow{\text{QFT}}$$
$$\frac{1}{2^{2\ell}} \cdot \sum_{a=0}^{2^\ell-1} \sum_{b=0}^{2^\ell-1} \sum_{j=0}^{2^\ell-1} \sum_{k=0}^{2^\ell-1} \mathrm{e}^{\,2\pi i\,(aj+bk)/2^\ell}\,|\,j, k, [a\ -\ bd]\,g\,\rangle\,.$$

4. Observe the system in a measurement to obtain (j, k) and $[e]\,g$.

Note that the size of the two first registers and the size of the QFT is determined by the order q of the group. The actual choice of group only affects the memory and time complexity of computing and storing $[a]\,g \odot [-b]\,x$.

Note furthermore that we assume in step 2 that the third register is of sufficient length in qubits to hold the result of the computation, that is the representation of elements of $\mathbb{G}$. We also assume that the group operation, and in particular the exponentiation operations, may be implemented efficiently using quantum circuits and that we have sufficient ancillary qubits as required to perform the computation.

The above algorithm is different from Shor's original algorithm in that it is described in general terms for a group $\mathbb{G}$ of prime order q, in that $\ell = \lfloor \log_2 q \rfloor$ and in that the algorithm initializes the system to the uniform distribution over all 2^ℓ states as opposed to a uniform distribution over q states.

4.1.1 Analysis of the probability distribution

Let $e = a - bd \bmod q$. When the system above is observed, the state $|\,j, k, [e]\,g\,\rangle$ is obtained with probability

$$\frac{1}{2^{4\ell}} \cdot \left| \sum_a \sum_b \exp\left[\frac{2\pi i}{2^\ell}\,(aj + bk)\right] \right|^2$$

where the sum is over all pairs (a, b) that produce this specific e.

1. Since $e = a - bd \bmod q$ we have $a = e + bd \bmod q$ so

$$aj + bk = ((e + bd) \bmod q)j + bk = ej + bdj - \left\lfloor \frac{e + bd}{q} \right\rfloor jq + bk$$

since $u \bmod q = u - \lfloor u/q \rfloor\, q$. Therefore the probability may be written

$$\frac{1}{2^{4\ell}} \cdot \left| \sum_b \exp\left[\frac{2\pi i}{2^\ell}\left(ej + b(dj + k) - \left\lfloor \frac{e + bd}{q} \right\rfloor jq\right)\right] \right|^2.$$

2. Extracting the term containing e yields

$$\frac{1}{2^{4\ell}} \cdot \underbrace{\left| \exp\left[\frac{2\pi i}{2^\ell} ej\right] \right|^2}_{=1} \cdot \left| \sum_b \exp\left[\frac{2\pi i}{2^\ell}\left(b(dj+k) - \left\lfloor \frac{e+bd}{q} \right\rfloor jq\right)\right] \right|^2 .$$

3. Centering b around zero yields

$$\frac{1}{2^{4\ell}} \cdot \left| \sum_b \exp\left[\frac{2\pi i}{2^\ell}\left(\left(b + 2^{\ell-1} - 2^{\ell-1}\right)(dj+k) - \left\lfloor \frac{e+bd}{q} \right\rfloor jq\right)\right] \right|^2 =$$

$$\frac{1}{2^{4\ell}} \cdot \underbrace{\left| e^{-\pi i(dj+k)} \right|^2}_{=1} \cdot \left| \sum_b \exp\left[\frac{2\pi i}{2^\ell}\left(\left(b - 2^{\ell-1}\right)(dj+k) - \left\lfloor \frac{e+bd}{q} \right\rfloor jq\right)\right] \right|^2 .$$

4. This probability may be written

$$\frac{1}{2^{4\ell}} \cdot \left| \sum_b \exp\left[\frac{2\pi i}{2^\ell}\left(\left(b - 2^{\ell-1}\right)(dj+k) - \left\lfloor \frac{e+bd}{q} \right\rfloor \{jq\}_{2^\ell}\right)\right] \right|^2$$

since adding or subtracting multiples of 2^ℓ has no effect; it is equivalent to shifting the phase angle by a multiple of 2π.

We will defer further analysis of the distribution to the proof of lemma 2 in section 4.1.4 and instead proceed to introduce the notion of a good pair.

4.1.2 The notion of a good pair (j, k)

In this section, we introduce the notion of a good pair. In the following sections, we will proceed to lower-bound the number of good pairs, to lower-bound the probability of obtaining a good pair from a single execution of the algorithm, and to demonstrate that d may be computed from a good pair.

It may not yet be apparent why we choose the below definition of a good pair but we will see shortly, when we reach the proof of lemma 2 in section 4.1.4, that there is a good rationale for selecting this particular definition.

Definition 1. *A pair (j, k) where j is an integer on the interval $0 \leq j < 2^\ell$ is said to be good if j is such that*

$$|\{jq\}_{2^\ell}| \leq 2^{\ell-4} \quad \textit{and} \quad \left| \{jq\}_{2^\ell} \frac{d}{q} - f_j \right| \leq 2^{-3} \quad \textit{for some} \quad f_j \in \mathbb{Z}$$

and $k \equiv f_j - dj \pmod{2^\ell}$.

Note that k is a function of j. Hence j is distinct for each good pair.

4.1.3 Lower-bounding the number of good pairs (j, k)

Lemma 1. *There are at least $2^{\ell-8} + 1$ good pairs (j, k).*

Proof. From definition 1 it follows that a pair (j, k) where j is an integer on the interval $0 \le j < 2^\ell$ and $k = f_j - dj \bmod 2^\ell$ is good if

$$|\,\{jq\}_{2^\ell}\,| \le 2^{\ell-4} \quad \text{and} \quad |\,\alpha\{jq\}_{2^\ell} - \lfloor \alpha\{jq\}_{2^\ell} \rceil\,| \le 2^{-3} \quad \text{where} \quad \alpha = \frac{d}{q} \in \mathbb{R}.$$

To lower-bound the number of j that meet this definition, consider the points

$$p_j = (x_j, y_j) = (\{jq\}_{2^\ell},\ \alpha\{jq\}_{2^\ell} \bmod 1)$$

plotted in a two-dimensional space for $0 \le j < 2^\ell$ where

$$-2^\ell/2 \le x_j < 2^\ell/2 \quad \text{and} \quad 0 \le y_j < 1$$

and partition the space into rectangles of side length $2^{\ell-4}$ in the x-direction and 2^{-3} in the y-direction. If p_{j_1} and p_{j_2} are any two points

$$\{j_2 q\}_{2^\ell} - \{j_1 q\}_{2^\ell} \equiv \{(j_2 - j_1)q\}_{2^\ell} \pmod{2^\ell}. \tag{1}$$

Furthermore, if p_{j_1} and p_{j_2} fall within the same rectangle

$$|\,\{j_2 q\}_{2^\ell} - \{j_1 q\}_{2^\ell}\,| \le 2^{\ell-4}. \tag{2}$$

As two numbers that are equal modulo 2^ℓ and both less than $2^{\ell-1}$ are in fact equal, equations (1) and (2) imply

$$|\,\{j_2 q\}_{2^\ell} - \{j_1 q\}_{2^\ell}\,| = |\,\{(j_2 - j_1)q\}_{2^\ell}\,| \le 2^{\ell-4}. \tag{3}$$

Analogously, if p_{j_1} and p_{j_2} are any two points

$$\begin{aligned}(\,\alpha\{j_2 q\}_{2^\ell} \bmod 1\,) - (\,\alpha\{j_1 q\}_{2^\ell} \bmod 1\,) \equiv \\ \alpha\,(\{j_2 q\}_{2^\ell} - \{j_1 q\}_{2^\ell}) \qquad \pmod 1.\end{aligned} \tag{4}$$

Furthermore, if p_{j_1} and p_{j_2} fall within the same rectangle

$$|\,(\,\alpha\{j_2 q\}_{2^\ell} \bmod 1\,) - (\,\alpha\{j_1 q\}_{2^\ell} \bmod 1\,)\,| \le 2^{-3}. \tag{5}$$

In conjunction (4) and (5) imply

$$\begin{aligned}|\,(\,\alpha\{j_2 q\}_{2^\ell} \bmod 1\,) - (\,\alpha\{j_1 q\}_{2^\ell} \bmod 1\,)\,| = \\ |\,\alpha(\{j_2 q\}_{2^\ell} - \{j_1 q\}_{2^\ell}) \bmod 1\,| \le 2^{-3}\end{aligned}$$

which by (3) reduces to

$$|\,\alpha\{(j_2 - j_1)q\}_{2^\ell} \bmod 1\,| \le 2^{-3}.$$

Hence, if p_{j_1} and p_{j_2} are in the same rectangle, with $j_2 \ge j_1$, and we let

$$j' = j_2 - j_1 \quad \text{and} \quad k' = (f_{j'} - dj') \bmod 2^\ell \quad \text{where} \quad f_{j'} = \lfloor \alpha\{j'q\}_{2^\ell} \rceil$$

then (j', k') is a good pair since $0 \le j' < 2^\ell$ and

$$|\,\{j'q\}_{2^\ell}\,| \le 2^{\ell-4} \quad \text{and} \quad |\,\alpha\{j'q\}_{2^\ell} - \lfloor \alpha\{j'q\}_{2^\ell} \rceil\,| \le 2^{-3}.$$

Our goal is now to count the number of points that fulfill these conditions.

Let R_i denote the number of points that fall into rectangle i. There are then

$$\frac{R_i\,(R_i+1)}{2} > \frac{R_i^2}{2} \tag{6}$$

pairwise combinations (p_{j_1}, p_{j_2}) of the R_i points in rectangle i such that $j_2 \geq j_1$. To lower-bound the number of good pairs (j,k), we therefore lower-bound

$$\frac{1}{2}\sum_{i=0}^{2^7-1} R_i^2$$

which is easy using the Cauchy-Schwarz inequality since

$$2^{2\ell} = \left(\sum_{i=0}^{2^7-1} R_i\right)^2 \leq \left(\sum_{i=0}^{2^7-1} 1^2\right)\left(\sum_{i=0}^{2^7-1} R_i^2\right) \quad \Rightarrow \quad \frac{1}{2}\sum_{i=0}^{2^7-1} R_i^2 \geq 2^{2\ell-8}$$

as there are $(2^\ell/2^{\ell-4}) \cdot (1/2^{-3}) = 2^7$ rectangles.

We have demonstrated that for two points p_{j_1} and p_{j_2} that fall within the same rectangle and that are such that $j_2 \geq j_1$ the pair (j', k') is a good. Furthermore, we have shown that there are more than $2^{2\ell-8}$ such pairs as the equality in (6) is strict. However, these pairs appear with multiplicity.

More specifically, if j_1 and $j_2 \geq j_1$ give rise to the good pair (j', k') then so do $j_1 + u$ and $j_2 + u$ for any integer u such that $0 \leq j_1 + u \leq j_2 + u < 2^\ell$ so each good pair is counted with multiplicity at most 2^ℓ.

From this observation it follows that there are more than $2^{2\ell-8}/2^\ell = 2^{\ell-8}$ distinct good pairs, and so the lemma follows. ■

4.1.4 Lower-bounding the probability of a good pair (j,k)

To lower-bound the probability of a good pair we first need to lower-bound the number of pairs (a,b) that yield a certain e.

Definition 2. *Let T_e denote the number of pairs (a,b) such that*

$$e \equiv a - bd \pmod{q}$$

where a, b are integers on the interval $0 \leq a, b < 2^\ell$.

Claim 2.

$$\sum_{e=0}^{q-1} T_e = 2^{2\ell}.$$

Proof. Since a, b may independently assume 2^ℓ values, there are $2^{2\ell}$ distinct pairs (a,b), from which the above claim follows. ■

Claim 3.

$$\sum_{e=0}^{q-1} T_e^2 \geq \frac{2^{4\ell}}{q}.$$

Proof. The claim follows from the Cauchy–Schwarz inequality and claim 2 since

$$2^{4\ell} = \left(\sum_{e=0}^{q-1} T_e\right)^2 \leq \left(\sum_{e=0}^{q-1} 1^2\right)\left(\sum_{e=0}^{q-1} T_e^2\right) \quad \Rightarrow \quad \sum_{e=0}^{q-1} T_e^2 \geq \frac{2^{4\ell}}{q}.$$

■

We are now ready to demonstrate a lower-bound the probability of obtaining a good pair using the above definition and claims.

Lemma 2. *The probability of obtaining a specific good pair (j,k) from a single execution of the algorithm in section 4.1 is at least*

$$\frac{1}{2q}.$$

Proof. The probability of observing the pair (j,k) and $[e]\,g$ is

$$\frac{1}{2^{4\ell}} \cdot \left| \sum_b \exp\left[\frac{2\pi i}{2^\ell}\left(\left(b-2^{\ell-1}\right)(dj+k) - \left\lfloor \frac{e+bd}{q} \right\rfloor \{jq\}_{2^\ell}\right)\right] \right|^2.$$

For a good pair $k = f_j - dj \bmod 2^\ell$ so the probability then reduces to

$$\frac{1}{2^{4\ell}} \cdot \left| \sum_b \exp\left[\frac{2\pi i}{2^\ell}\left(\left(b-2^{\ell-1}\right) f_j - \left\lfloor \frac{e+bd}{q} \right\rfloor \{jq\}_{2^\ell}\right)\right] \right|^2.$$

If an additional constant term is inserted to shift the global phase

$$\frac{1}{2^{4\ell}} \cdot \underbrace{\left| \mathrm{e}^{\pi i \{jq\}_{2^\ell} \frac{d}{q}} \right|^2}_{=1} \cdot \left| \sum_b \exp\left[\frac{2\pi i}{2^\ell}\left((b-2^{\ell-1}) f_j - \left\lfloor \frac{e+bd}{q} \right\rfloor \{jq\}_{2^\ell}\right)\right] \right|^2 =$$

$$\frac{1}{2^{4\ell}} \cdot \left| \sum_b \exp\left[\frac{2\pi i}{2^\ell}\left((b-2^{\ell-1}) f_j - \left\lfloor \frac{e+bd}{q} \right\rfloor \{jq\}_{2^\ell} + 2^{\ell-1}\{jq\}_{2^\ell}\frac{d}{q}\right)\right] \right|^2$$

then by the triangle inequality

$$\left| \{jq\}_{2^\ell} \left\lfloor \frac{e+bd}{q} \right\rfloor - (b-2^{\ell-1}) f_j - 2^{\ell-1}\{jq\}_{2^\ell}\frac{d}{q} \right| \leq$$
$$\left| \{jq\}_{2^\ell} \left(\left\lfloor \frac{e+bd}{q} \right\rfloor - \frac{bd}{q} \right) \right| + \left| (b-2^{\ell-1}) \left(\{jq\}_{2^\ell}\frac{d}{q} - f_j \right) \right|.$$

Since $e = a - bd \bmod q$ we have that $0 \leq e < q$ which implies

$$\frac{bd}{q} - 1 \leq \left\lfloor \frac{e+bd}{q} \right\rfloor \leq \frac{bd}{q} + 1 \quad \Rightarrow \quad \left| \left\lfloor \frac{e+bd}{q} \right\rfloor - \frac{bd}{q} \right| \leq 1.$$

Furthermore, for a good pair $|\,\{jq\}_{2^\ell}\,| \leq 2^{\ell-4}$ and

$$\left| \{jq\}_{2^\ell}\frac{d}{q} - f_j \right| \leq 2^{-3} \quad \Rightarrow \quad \left| (b-2^{\ell-1}) \left(\{jq\}_{2^\ell}\frac{d}{q} - f_j \right) \right| \leq 2^{\ell-4}$$

where we have used that $0 \leq b < 2^{\ell}$. Combining the above results yields

$$\left| \{jq\}_{2^\ell} \left\lfloor \frac{e+bd}{q} \right\rfloor - (b - 2^{\ell-1}) f_j - 2^{\ell-1} \{jq\}_{2^\ell} \frac{d}{q} \right| \leq |\, \{jq\}_{2^\ell} \,| + 2^{\ell-4} \leq 2^{\ell-3}$$

It therefore follows from claim 1 in section 3.5.2 that the sum

$$\frac{1}{2^{4\ell}} \cdot \left| \sum_b \exp\left[\frac{2\pi i}{2^\ell} \left(b f_j - \left\lfloor \frac{e+bd}{q} \right\rfloor \{jq\}_{2^\ell} \right) \right] \right|^2 = \frac{1}{2^{4\ell}} \cdot \left| \sum_b \mathrm{e}^{i\theta_b} \right|^2 \geq \frac{T_e^2}{2 \cdot 2^{4 \cdot \ell}}$$

since $|\, \theta_b \,| \leq 2\pi/8 = \pi/4$ for all T_e values of b. Summing over all e, we obtain

$$\sum_{e=0}^{q-1} \frac{T_e^2}{2 \cdot 2^{4\ell}} \geq \frac{2^{4\ell}}{q} \cdot \frac{1}{2 \cdot 2^{4\ell}} = \frac{1}{2q}$$

by claim 3 and so the lemma follows. ■

We note that the proof of lemma 2, although developed independently, bears some resemblance to a proof in the generalization by Boneh and Lipton [1].

4.2 Computing d from a good pair (j, k)

In this section, we specify a classical algorithm that upon input of a good pair (j, k), that results from the execution of the quantum algorithm in section 4.1 with g and $x = [d]\, g$ as input, computes and outputs d assuming $j \neq 0$.

1. Compute and return

$$d \equiv \left\lfloor \frac{kq}{2^\ell} \right\rceil t^{-1} \pmod{q} \quad \text{where} \quad t = \frac{\{jq\}_{2^\ell} - jq}{2^\ell} \in \mathbb{Z}.$$

Note that this is exactly the same algorithm as was originally proposed by Shor.

4.2.1 Proof of correctness

In this section we provide a proof of correctness for the above algorithm.

Lemma 3. *For any good pair (j, k) such that $j \neq 0$ it holds that*

$$d \equiv \left\lfloor \frac{kq}{2^\ell} \right\rceil t^{-1} \pmod{q} \quad \textit{where} \quad t = \frac{\{jq\}_{2^\ell} - jq}{2^\ell} \in \mathbb{Z}.$$

Proof. For a good pair we have by definition 1 that

$$\left| \{jq\}_{2^\ell} \frac{d}{q} - f_j \right| \leq 2^{-3} \quad \Rightarrow \quad \{jq\}_{2^\ell} \frac{d}{q} - f_j = \delta$$

where $|\, \delta \,| \leq 2^{-3}$ is an error term, and furthermore

$$k \equiv f_j - dj \pmod{2^l}$$

which yields

$$\{jq\}_{2^\ell} \frac{d}{q} - k - dj + n2^l = \delta.$$

Multiplying by q yields

$$\{jq\}_{2^\ell} d - kq - djq + nq2^l = \delta q.$$

Re-arranging terms and dividing by 2^ℓ yields

$$d \underbrace{\frac{\{jq\}_{2^\ell} - jq}{2^\ell}}_{t \in \mathbb{Z}} - \frac{kq}{2^\ell} + nq = \frac{\delta q}{2^\ell}.$$

Rounding both sides to the nearest integer yields

$$dt + \left\lfloor -\frac{kq}{2^\ell} \right\rceil + nq = \left\lfloor dt - \frac{kq}{2^\ell} + nq \right\rceil = \left\lfloor \frac{\delta q}{2^\ell} \right\rceil = 0$$

and hence

$$dt + \left\lfloor -\frac{kq}{2^\ell} \right\rceil \equiv 0 \pmod{q}$$

from which the result in the lemma follows

$$d \equiv \left\lfloor \frac{kq}{2^\ell} \right\rceil t^{-1} \pmod{q}$$

provided that $t \not\equiv 0 \pmod{q}$. Since $j > 0$ we have $jq > 2^\ell$ and $|\{jq\}_{2^\ell}| \leq 2^\ell/2$ which implies $0 < |t|$. Furthermore

$$|t| = \left| \frac{\{jq\}_{2^\ell} - jq}{2^\ell} \right| \leq \frac{1}{2} + \frac{jq}{2^\ell} \leq \frac{1}{2} + \frac{(2^\ell - 1)q}{2^\ell} \leq \frac{1}{2} + q - \underbrace{\frac{q}{2^\ell}}_{>1} < q.$$

Hence $0 < |t| < q$ which implies $t \not\equiv 0 \pmod{q}$ and so the lemma follows. ■

4.3 Main result

Theorem 1 summarizes the main result in this section.

Theorem 1. *Assume that the algorithm in section 4.1 is executed once on a quantum computer with g and $x = [d]\,g$ as input, and that it yields some pair (j, k) as output. If this pair (j, k) is then fed as input to the classical algorithm in section 4.2, it will output d with probability at least 2^{-10}.*

Proof. By lemma 2 the probability of obtaining a specific good pair (j, k) as a result of a single execution of the quantum algorithm in section 4.1 is at least $1/(2q)$. By lemma 1 there are at least $2^{\ell-8} + 1$ good pairs (j, k). By definition 1 the value of j is distinct for each good pair.

It therefore follows from lemmas 1 and 2 and from definition 1 that the probability of obtaining some good pair (j, k) such that $j \neq 0$ must be at least

$$2^{\ell-8} \cdot \frac{1}{2q} \geq 2^{-10}$$

where we have used that $2^\ell/q \geq 1/2$.

By lemma 3 the classical algorithm in section 4.2 returns d for all good pairs (j, k) such that $j \neq 0$ and so the theorem follows. ■

It should be stressed that we have have not sought the best lower bound on the success probability in this paper. Rather we have sought a lower bound that is easy to prove and that is sufficiently good to show that the algorithm is practical. We expect the actual success probability to be much greater than what is indicated by the bound.

5 Computing d in the special case where $d \lll q$

In this section, we describe how the algorithm in section 4 may modified to be more efficient in the special case where $d \lll q$.

More specifically, we consider the special case where $q \geq 2^{2\ell} + (2^{\ell} - 1)d$ and where d is on the interval $0 < d < 2^{\ell}$ for some integer ℓ. Note that ℓ is now different from ℓ in section 4. As in section 4, the algorithm may be said to consist of a quantum part and a classical part.

- The quantum algorithm is described in section 5.1.

  It accepts as input a generator g and an element $x = [d]\, g$ and produces as output a pair (j, k) and an element $[e]\, g$ that is ignored.

  The algorithm in section 5.1 is similar to the algorithm in section 4.1.

  - The two index registers are however now of length ℓ and 2ℓ qubits, compared to both being of length $\lfloor \log_2 q \rfloor$ qubits in section 4.1.
  - Consequently, when $[a]\, g \odot [-b]\, x$ is now computed $0 \leq a < 2^{2\ell}$ and $0 \leq b < 2^{\ell}$ compared to $0 \leq a, b < 2^{\lfloor \log_2 q \rfloor}$ in section 4.1.
  - Furthermore, the two QFTs are now of size 2^{ℓ} and $2^{2\ell}$, compared to both being of size $2^{\lfloor \log_2 q \rfloor}$ in section 4.1.

- The classical algorithm is described in section 5.2.

  It accepts as input the pair (j, k) output by the quantum algorithm and yields the discrete logarithm d if the pair (j, k) is good and if it fulfills certain additional criteria that are elaborated on in section 5.2.

  The algorithm in section 5.2 is quite different from the algorithm in section 4.2 in that is uses lattice-based techniques to recover d from (j, k).

5.1 The quantum algorithm for computing (j, k)

In this section we provide a detailed description of the quantum algorithm that upon input of g and $x = [d]\, g$ outputs a pair (j, k).

1. Let

$$|\, \Psi \,\rangle = \frac{1}{\sqrt{2^{3\ell}}} \sum_{a\,=\,0}^{2^{2\ell}-1} \sum_{b\,=\,0}^{2^{\ell}-1} |\, a \,\rangle\, |\, b \,\rangle\, |\, 0 \,\rangle$$

   where the first and second registers are of length ℓ and 2ℓ qubits.

2. Compute $[a]\,g \odot [-b]\,x$ and store the result in the third register

$$\begin{aligned} |\,\Psi\,\rangle &= \frac{1}{\sqrt{2^{3\ell}}} \sum_{a=0}^{2^{2\ell}-1} \sum_{b=0}^{2^{\ell}-1} |\,a, b, [a]\,g \odot [-b]\,x\,\rangle \\ &= \frac{1}{\sqrt{2^{3\ell}}} \sum_{a=0}^{2^{2\ell}-1} \sum_{b=0}^{2^{\ell}-1} |\,a, b, [a\ -\ bd]\,g\,\rangle\,. \end{aligned}$$

3. Compute a QFT of size $2^{2\ell}$ of the first register and a QFT of size 2^{ℓ} of the second register to obtain

$$\begin{aligned} |\,\Psi\,\rangle = \frac{1}{\sqrt{2^{3\ell}}} \sum_{a=0}^{2^{2\ell}-1} \sum_{b=0}^{2^{\ell}-1} |\,a, b, [a\ -\ bd]\,g\,\rangle \quad \xrightarrow{\text{QFT}} \\ \frac{1}{\sqrt{2^{6\ell}}} \sum_{a=0}^{2^{2\ell}-1} \sum_{b=0}^{2^{\ell}-1} \sum_{j=0}^{2^{2\ell}-1} \sum_{k=0}^{2^{\ell}-1} \mathrm{e}^{\,2\pi i\,(aj+2^{\ell}bk)/2^{2\ell}}\,|\,j, k, [a\ -\ bd]\,g\,\rangle\,. \end{aligned}$$

4. Observe the system in a measurement to obtain (j, k) and $[e]\,g$.

Note that the size of the two first registers and the size of the QFT is now determined by d and not by q as was the case in section 4.1.

5.1.1 Analysis of the probability distribution

When the system above is observed, the state $|\,j, k, [e]\,g\,\rangle$, where $e = a - bd$, is obtained with probability

$$\frac{1}{2^{6\ell}} \cdot \left| \sum_{a} \sum_{b} \exp\left[\frac{2\pi i}{2^{2\ell}}\,(aj + 2^{\ell}bk)\right] \right|^2$$

where the sum is over all pairs (a, b) that produce this specific e. Note that the assumption that $q \geq 2^{2\ell} + (2^{\ell} - 1)d$ implies that no reduction modulo q occurs when e is computed. In what follows, we re-write the above expression for the probability on a form that is easier to use in practice.

1. Since $e = a - bd$ we have $a = e + bd$ so the probability may be written

$$\frac{1}{2^{6\ell}} \cdot \left| \sum_{b} \exp\left[\frac{2\pi i}{2^{2\ell}}\,((e + bd)j + 2^{\ell}bk)\right] \right|^2$$

   where the sum is over all b in the set $\{\,0 \leq b < 2^{\ell} \mid 0 \leq a = e + bd < 2^{2\ell}\,\}$.

2. Extracting the term containing e yields

$$\frac{1}{2^{6\ell}} \cdot \underbrace{\left| \exp\left[\frac{2\pi i}{2^{2\ell}}\,ej\right] \right|^2}_{=1} \cdot \left| \sum_{b} \exp\left[\frac{2\pi i}{2^{2\ell}}\,b\,(dj + 2^{\ell}k)\right] \right|^2.$$

3. Centering b around zero yields

$$\frac{1}{2^{6\ell}} \cdot \left| \sum_b \exp\left[\frac{2\pi i}{2^{2\ell}} \left(b + 2^{\ell-1} - 2^{\ell-1}\right)(dj + 2^\ell k)\right] \right|^2 =$$

$$\frac{1}{2^{6\ell}} \cdot \underbrace{\left| \exp\left[\frac{\pi i}{2^\ell}(dj + 2^\ell k)\right] \right|^2}_{=1} \cdot \left| \sum_b \exp\left[\frac{2\pi i}{2^{2\ell}} \left(b - 2^{\ell-1}\right)(dj + 2^\ell k)\right] \right|^2 .$$

4. This probability may be written

$$\frac{1}{2^{6\ell}} \cdot \left| \sum_b \exp\left[\frac{2\pi i}{2^{2\ell}} \left(b - 2^{\ell-1}\right) \{dj + 2^\ell k\}_{2^{2\ell}}\right] \right|^2$$

since adding or subtracting multiples of $2^{2\ell}$ has no effect; it is equivalent to shifting the phase angle by a multiple of 2π.

5.1.2 The notion of a good pair (j, k)

By claim 1 this sum should be large when $|\{dj + 2^\ell k\}_{2^{2\ell}}| \leq 2^{\ell-2}$ since this condition implies that the angle is less than or equal to $\pi/4$.

This observation serves as our motivation for introducing the below notion of a good pair, and for proceeding in the following sections to lower-bound the number of good pairs and the probability of obtaining a specific good pair.

Definition 3. *A pair (j, k), where j is an integer such that $0 \leq j < 2^{2\ell}$ and $k = -\lfloor (dj \bmod 2^{2\ell})/2^\ell \rceil \bmod 2^\ell$ is said to be good if*

$$\left| \{dj + 2^\ell k\}_{2^{2\ell}} \right| \leq 2^{\ell-2}.$$

Note that k is a function of j. Hence j is distinct for each good pair.

5.1.3 Lower-bounding the number of good pairs (j, k)

Lemma 4. *There are at least $2^{2\ell-1}$ good pairs.*

Proof. First note that for a good pair

$$\left| \{dj + 2^\ell k\}_{2^{2\ell}} \right| = |\{dj\}_{2^\ell}| \leq 2^{\ell-2}$$

since $k = -\lfloor (dj \bmod 2^{2\ell})/2^\ell \rceil \bmod 2^\ell$.

Let 2^κ be the greatest power of two that divides d. Since $d < 2^\ell$ it must be that $\kappa \leq \ell - 1$. As j runs through all integers $0 \leq j < 2^{2\ell}$, the function $dj \bmod 2^\ell$ assumes the value of each multiple of 2^κ exactly $2^{\ell+\kappa}$ times.

Assume that $\kappa = \ell - 1$. Then the only possible values are 0 and $2^{\ell-1}$. Only zero gives rise to a good pair, so with multiplicity there are $2^{\ell+\kappa} = 2^{2\ell-1}$ integers j such that (j, k) is a good pair.

Assume that $\kappa < \ell - 1$. Then only the $2 \cdot 2^{\ell-\kappa-2} + 1$ values congruent to values on $[-2^{\ell-2}, 2^{\ell-2}]$ are such that $|\{dj\}_{2^\ell}| \leq 2^{\ell-2}$, so with multiplicity $2^{\ell+\kappa}$ there are $2^{\ell+\kappa} \cdot (2 \cdot 2^{\ell-\kappa-2} + 1) \geq 2^{2\ell-1}$ integers j such that (j, k) is a good pair. ■

5.1.4 Lower-bounding the probability of a good pair (j, k)

To lower-bound the probability of a good pair we first need to lower-bound the number of pairs (a, b) that yield a certain e.

Definition 4. *Let T_e denote the number of pairs (a, b) such that*

$$e = a - bd$$

where a, b are integers on the intervals $0 \leq a < 2^{2\ell}$ and $0 \leq b < 2^{\ell}$.

Claim 4.

$$|\, e = a - bd \,| < 2^{2\ell}$$

Proof. The claim follows from $0 \leq a < 2^{2\ell}$, $0 \leq b < 2^{\ell}$ and $0 < d < 2^{\ell}$. ■

Claim 5.

$$\sum_{e=-2^{2\ell}}^{2^{2\ell}-1} T_e = 2^{3\ell}.$$

Proof. Since a, b may independently assume $2^{2\ell}$ and 2^{ℓ} values respectively, there are $2^{3\ell}$ distinct pairs (a, b). From this fact and claim 4 the claim follows. ■

Claim 6.

$$\sum_{e=-2^{2\ell}}^{2^{2\ell}-1} T_e^2 \geq 2^{4\ell-1}.$$

Proof. The claim follows from the Cauchy–Schwarz inequality and claim 5 since

$$2^{6\ell} = \left(\sum_{e=-2^{2\ell}}^{2^{2\ell}-1} T_e \right)^2 \leq \left(\sum_{e=-2^{2\ell}}^{2^{2\ell}-1} 1^2 \right) \left(\sum_{e=-2^{2\ell}}^{2^{2\ell}-1} T_e^2 \right) \quad \Rightarrow \quad \sum_{e=-2^{2\ell}}^{2^{2\ell}-1} T_e^2 \geq 2^{4\ell-1}.$$

■

We are now ready to demonstrate a lower-bound on the probability of obtaining a good pair using the above definition and claims.

Lemma 5. *The probability of obtaining a specific good pair (j, k) from a single execution of the algorithm in section 5.1 is at least $2^{-2\ell-2}$.*

Proof. For a good pair

$$\left| \frac{2\pi}{2^{2\ell}} \left(b - 2^{\ell-1}\right) \{dj + 2^{\ell} k\}_{2^{2\ell}} \right| \leq \frac{2\pi}{2^{\ell+2}} \; |\, b - 2^{\ell-1} \,| \leq \frac{\pi}{4}$$

for any integer b on the interval $0 \leq b < 2^{\ell}$. It therefore follows from claim 1 in section 3.5.2 that the probability

$$\frac{1}{2^{6\ell}} \cdot \left| \sum_{b} \exp\left[\frac{2\pi i}{2^{2\ell}} \left(b - 2^{\ell-1}\right) \{dj + 2^{\ell} k\}_{2^{2\ell}} \right] \right|^2 \geq \frac{T_e^2}{2 \cdot 2^{6\ell}}$$

for a specific e. Summing this over all e and using claim 6 yields

$$\sum_{e=-2^{2\ell}}^{2^{2\ell}-1} \frac{T_e^2}{2 \cdot 2^{6\ell}} \geq 2^{-2\ell-2}$$

from which the lemma follows. ■

5.2 Computing d from a good pair (j, k)

In this section, we specify a classical algorithm that upon input of a good pair (j, k), that results from a single execution of the algorithm in section 5.1 with g and $x = [d]\, g$ as input, computes and outputs d.

The algorithm uses lattice-based techniques. To introduce the algorithm, we first need to define the lattice $L(j)$.

Definition 5. *Let $L(j)$ be the integer lattice generated by the row span of*

$$\begin{bmatrix} 4j & 1 \\ 4 \cdot 2^{2\ell} & 0 \end{bmatrix}.$$

The algorithm proceeds as follows to recover d from (j, k) using $L(j)$.

1. Let $\vec{v} = (4\,\{-2^\ell k\}_{2^{2\ell}},\, 0) \in \mathbb{Z}^2$. For all vectors $\vec{u} \in L(j)$ such that

   $$|\,\vec{u} - \vec{v}\,| < \sqrt{2} \cdot 2^\ell$$

   test if the second component of $\vec{u}$ is d. If so return d.

   This test may be performed by checking if $x = [d]\, g$.

2. If d is not found in step 1, or when more than 23 vectors have been explored, the algorithm fails.

5.2.1 Rationale and analysis

For any $m \in \mathbb{Z}$ the vector $\vec{u} = (4\,(\{dj\}_{2^{2\ell}} + m2^{2\ell}),\, d) \in L(j)$. The above algorithm performs an exhaustive search of all vectors in $L(j)$ at distance at most $\sqrt{2} \cdot 2^\ell$ from $\vec{v}$ to find $\vec{u}$ for some m. It then recovers d as the second component of $\vec{u}$. The search will succeed in finding $\vec{u}$ if

$$\begin{aligned} |\,\vec{u} - \vec{v}\,| &= \sqrt{4\,(\{dj\}_{2^{2\ell}} + m\,2^{2\ell} - \{-2^\ell k\}_{2^{2\ell}})^2 + d^2} \\ &= \sqrt{4\,(\{dj + 2^\ell k\}_{2^{2\ell}})^2 + d^2} < \sqrt{2} \cdot 2^\ell \end{aligned}$$

since $d < 2^\ell$ and $|\{dj + 2^\ell k\}_{2^{2\ell}}| \leq 2^{\ell-2}$ by the definition of a good pair, and since m may be freely selected to obtain equality.

Whether the search is computationally feasible depends on the number of vectors in $L(j)$ that lie within distance $\sqrt{2} \cdot 2^\ell$ of $\vec{v}$. Lemma 6 below relates this number to the norm of the shortest non-zero vector in the lattice. To introduce the lemma, we first need some additional details.

Definition 6. *Let $\vec{s}_1$ be the shortest non-zero vector in $L(j)$, and let $\vec{s}_2$ be the shortest non-zero vector in $L(j)$ that is linearly independent to $\vec{s}_1$.*

The vectors $\vec{s}_1$ and $\vec{s}_2$ form a reduced basis for $L(j)$. Such a basis may be computed using standard lattice basis reduction algorithms such as Lenstra-Lenstra-Lováz (LLL) [5].

Definition 7. *Let $\vec{s}_2^{\,/\!/}$ be the component of $\vec{s}_2$ parallel to $\vec{s}_1$ and let $\vec{s}_2^{\,\perp}$ be the component of $\vec{s}_2$ orthogonal to $\vec{s}_1$.*

Claim 7. *The angle α between $\vec{s}_1$ and $\vec{s}_2$ is such that $\frac{\pi}{3} \leq \alpha \leq \frac{2\pi}{3}$.*

Proof. From definition 6, it follows that $|\,\vec{s}_2^{\,/\!/}\,| \leq |\,\vec{s}_1\,|\,/\,2$ and $|\,\vec{s}_1\,| \leq |\,\vec{s}_2\,|$, so

$$|\,\vec{s}_2^{\,/\!/}\,| = |\,\vec{s}_2\,| \cdot |\cos\alpha\,| \leq |\,\vec{s}_1\,|\,/2 \leq |\,\vec{s}_2\,|\,/2 \quad \Rightarrow \quad |\cos\alpha\,| \leq \frac{1}{2}$$

from which the claim follows, by solving for α on the interval $0 \leq \alpha \leq \pi$. ■

Lemma 6. *For $\vec{v} \in \mathbb{R}^2$ the number of lattice vectors $\vec{u} \in L(j)$ within distance $\sqrt{2} \cdot 2^\ell$ of $\vec{v}$ is lower-bounded by*

$$\max\left(12,\ \left\lfloor \frac{2\sqrt{2} \cdot 2^\ell}{|\,\vec{s}_1\,|} \right\rfloor + 1\right).$$

Proof. Any vector in $L(j)$ may be written on the form $i_1\vec{s}_1 + i_2\vec{s}_2$ where $i_1, i_2 \in \mathbb{Z}$. For fixed $i_2 \in \mathbb{Z}$ and variable $i_1 \in \mathbb{Z}$ the vectors $i_1\vec{s}_1 + i_2\vec{s}_2 \in L(j)$ are on a line l in $\mathbb{R}^2$. The number of vectors on l in any circle with radius $\sqrt{2} \cdot 2^\ell$ is at most

$$\left\lfloor \frac{2\sqrt{2} \cdot 2^\ell}{|\,\vec{s}_1\,|} \right\rfloor + 1.$$

The orthogonal distance between two consecutive lines on the above form is $|\,s_2^\perp\,|$. The vector $\vec{v} \in \mathbb{R}^2$ may be written on the form $\vec{v} = a_1\vec{s}_1 + a_2\vec{s}_2$ for some $a_1, a_2 \in \mathbb{R}$. For l to at all intersect the circle of radius $\sqrt{2} \cdot 2^\ell$ around v it must therefore be that $|\,i_2 - a_2\,| \cdot |\vec{s}_2^{\perp}| \leq \sqrt{2} \cdot 2^\ell$ and this is the case for at most

$$\left\lfloor \frac{2\sqrt{2} \cdot 2^\ell}{|\,\vec{s}_2^{\perp}\,|} \right\rfloor + 1.$$

different i_2. Hence, there are at most

$$S = \left(\left\lfloor \frac{2\sqrt{2} \cdot 2^\ell}{|\,\vec{s}_1\,|} \right\rfloor + 1\right)\left(\left\lfloor \frac{2\sqrt{2} \cdot 2^\ell}{|\,\vec{s}_2^{\perp}\,|} \right\rfloor + 1\right) \tag{7}$$

vectors in $L(j)$ within distance $\sqrt{2} \cdot 2^\ell$ of $\vec{v}$.

From claim 7 it follows that the angle α between $\vec{s}_1$ and $\vec{s}_2$ is such that

$$\frac{\pi}{3} \leq \alpha \leq \frac{2\pi}{3} \quad \Rightarrow \quad |\,\vec{s}_2^{\perp}\,| = |\,\vec{s}_1\,| \sin\alpha \geq \frac{\sqrt{3}}{2}\,|\,\vec{s}_1\,|$$

which inserted into equation (7) yields

$$S \leq \left(\left\lfloor \frac{2\sqrt{2} \cdot 2^\ell}{|\,\vec{s}_1\,|} \right\rfloor + 1\right)\left(\left\lfloor \sqrt{\frac{2}{3}}\,\frac{4 \cdot 2^\ell}{|\,\vec{s}_1\,|} \right\rfloor + 1\right). \tag{8}$$

Furthermore, the area of the fundamental parallelogram in $L(j)$ is

$$|\,\vec{s}_1\,| \cdot |\,\vec{s}_2^{\,\perp}\,| = |\,\det L(j)\,| = 4 \cdot 2^{2\ell} \;\;\Rightarrow\;\; |\,\vec{s}_2^{\,\perp}\,| = 4 \cdot 2^{2\ell} \,/\, |\,\vec{s}_1\,|$$

which inserted into equation (7) yields

$$S = \left(\left\lfloor \frac{2\sqrt{2} \cdot 2^{\ell}}{|\,\vec{s}_1\,|} \right\rfloor + 1 \right) \left(\left\lfloor \frac{|\,\vec{s}_1\,|}{\sqrt{2} \cdot 2^{\ell}} \right\rfloor + 1 \right). \tag{9}$$

Assume that $|\,\vec{s}_1\,| > 2^{\ell}$. Then, equation (8) implies $S \leq 12$. Conversely, assume that $|\,\vec{s}_1\,| \leq 2^{\ell}$. Then, equation (9) implies

$$S = \left\lfloor \frac{2\sqrt{2} \cdot 2^{\ell}}{|\,\vec{s}_1\,|} \right\rfloor + 1$$

and so the lemma follows by taking the maximum. ■

To perform the search in practice, all vectors on the form $i_1\vec{s}_1 + i_2\vec{s}_2$ where $i_1, i_2 \in \mathbb{Z}$ that lie within distance of $\sqrt{2} \cdot 2^{\ell}$ of $\vec{v}$ are exhaustively searched. The starting point is easily found by writing $\vec{v} = a_1\vec{s}_1 + a_2\vec{s}_2$ where $a_1, a_2 \in \mathbb{R}$ and rounding a_1, a_2 to the nearest integers.

Basis reduction algorithms such as LLL are practical for lattices of dimension two so it is practical to compute $\vec{s}_1$ and $\vec{s}_2$. Therefore the above algorithm is practical if $|\,\vec{s}_1\,|$ is large in relation to $\sqrt{2} \cdot 2^{\ell}$ so that the search space is small. To demonstrate that the algorithm is practical we lower-bound $|\,\vec{s}_1\,|$ by first introducing the notion of good integers j.

Definition 8. *An integer j on the interval $0 \leq j < 2^{2\ell}$ is said to be bad if there exists an integer ξ on the interval $0 < \xi < 2^{\ell-3}$ such that $|\,\{\xi j\}_{2^{2\ell}}\,| < 2^{\ell-5}$. Otherwise, j is said to be good.*

Lemma 7. *If j is good then the shortest non-zero vector $\vec{s}_1 \in L(j)$ has norm*

$$|\,\vec{s}_1\,| \geq 2^{\ell-3}.$$

Proof. The shortest non-zero vector $\vec{s}_1 \in L(j)$ may be written on the form

$$\vec{s}_1 = \begin{bmatrix} \xi & \lambda \end{bmatrix} \begin{bmatrix} 4j & 1 \\ 4 \cdot 2^{2\ell} & 0 \end{bmatrix} = \begin{bmatrix} 4\,(\xi j + \lambda 2^{2\ell}) & \xi \end{bmatrix} = \begin{bmatrix} 4\,\{\xi j\}_{2^{2\ell}} & \xi \end{bmatrix}$$

where λ must be used to reduce ξj modulo $2^{2\ell}$ constrained to the interval $-2^{2\ell}/2 \leq \{\xi j\}_{2^{2\ell}} < 2^{2\ell}/2$ so that $|\,4\,\{\xi j\}_{2^{2\ell}}\,|$ is minimized. Hence

$$|\,\vec{s}_1\,| = \sqrt{(4\,\{\xi j\}_{2^{2\ell}})^2 + \xi^2}$$

which implies that $|\,\vec{s}_1\,| \geq |\,4\,\{\xi j\}_{2^{2\ell}}\,|$ and that $|\,\vec{s}_1\,| \geq |\,\xi\,|$. Since $|\,\{u\}_{2^{2\ell}}\,| = |\,\{-u\}_{2^{2\ell}}\,|$ for all $u \in \mathbb{Z}$ we only consider positive ξ. If j is good then $|\,\{\xi j\}_{2^{2\ell}}\,| \geq 2^{\ell-5}$ for all $0 < \xi < 2^{\ell-3}$ by definition 8 and so the lemma follows. ■

This implies that if j is good then the exhaustive search needs explore only a small number of vectors to recover $\vec{u}$ and hence d.

5.2.2 Proof of correctness

Lemma 8 summarizes the above analysis and provides a proof of correctness.

Lemma 8. *Assume that the algorithm in section 5.1 is executed once on a quantum computer with g and $x = [d]\, g$ as input, and that it yields some good pair (j, k) as output that is such that the integer j is good. If this pair (j, k) is fed as input to the classical algorithm in section 5.2 it will output d.*

Proof. It follows from the analysis in section 5.2.1 that if the algorithm exhausts all vectors within distance $\sqrt{2} \cdot 2^{\ell}$ of $\vec{v}$ it will recover $\vec{u}$ and hence d. The search is practical if there are not too many vectors within distance $\sqrt{2} \cdot 2^{\ell}$ of $\vec{v}$.

By lemma 6 there are at most

$$\max\left(12,\ \left\lfloor \frac{2\sqrt{2} \cdot 2^{\ell}}{|\, \vec{s}_1 \,|} \right\rfloor + 1\right) \leq \max\left(12,\ \left\lfloor 2\sqrt{2} \cdot 2^{3} \right\rfloor + 1\right) = 23$$

vectors within distance $\sqrt{2} \cdot 2^{\ell}$ of $\vec{v}$ where we have used that by lemma 7 the norm $|\, s_1 \,| \geq 2^{\ell-3}$ for good j. As per the specification, the algorithm will not fail before having exhausted all 23 vectors, and so the lemma follows. ■

5.2.3 Upper-bounding the number of bad integers j

As a final step, we need to demonstrate an upper bound on the number of bad j, as this bound is needed to demonstrate a lower bound on the overall success probability of the algorithm in section 5.3.

Claim 8. *For a fixed ξ on the interval $0 < \xi < 2^{\ell-3}$ there are less than $2^{\ell-4}$ integers j on the interval $0 \leq j < 2^{2\ell}$ such that $|\, \{\xi j\}_{2^{2\ell}} \,| < 2^{\ell-5}$.*

Proof. Let 2^{κ} be the largest power of two that divides ξ. Since $\xi < 2^{\ell-3}$ it must be that $\kappa < \ell - 3$. As j runs through all integers $0 \leq j < 2^{2\ell}$, the function $\xi j \bmod 2^{2\ell}$ assumes the value of each multiple of 2^{κ} exactly $2^{2\ell-\kappa}$ times.

Only the $2 \cdot 2^{\ell-\kappa-5} - 1$ values congruent to values on $(-2^{\ell-5}, 2^{\ell-5})$ are such that $|\, \{\xi j\}_{2^{2\ell}} \,| < 2^{\ell-5}$ so with multiplicity 2^{κ} there are $2^{\kappa} \cdot (2 \cdot 2^{\ell-\kappa-5} - 1) < 2^{\ell-4}$ integers j such that $|\, \{\xi j\}_{2^{2\ell}} \,| < 2^{\ell-5}$ for a fixed ξ. ■

Lemma 9. *There are less than $2^{2\ell-7}$ bad integers j.*

Proof. By definition 8 there are $2^{\ell-3} - 1$ values of ξ since $0 < \xi < 2^{\ell-3}$ and by claim 8 there are less than $2^{\ell-4}$ bad integers j for a fixed ξ. In total, there are therefore less than $(2^{\ell-3} - 1) \cdot 2^{\ell-4} < 2^{2\ell-7}$ bad integers j. ■

5.3 Main result

Theorem 2 summarizes the main result in this section.

Theorem 2. *Assume that the algorithm in section 5.1 is executed once on a quantum computer with g and $x = [d]\, g$ as input, and that it yields some pair (j, k) as output. If this pair (j, k) is fed as input to the classical algorithm in section 5.2, it will output d with probability at least $2^{-3} \cdot (1 - 2^{-6})$.*

Proof. By lemma 4 there are at least $2^{2\ell-1}$ good pairs (j,k). It follows from definition 3 of a good pair that j is distinct. By lemma 9 there are less than $2^{2\ell-7}$ bad j. It follows that there must be at least $2^{2\ell-1}-2^{2\ell-7}$ good pairs (j,k) for which j is good.

By lemma 5 the probability of obtaining a specific good pair from a single execution of the algorithm in section 5.1 is at least $2^{-2\ell-2}$. The probability of obtaining a good pair (j,k) for which j is good must therefore be at least

$$2^{-2\ell-2} \cdot (2^{2\ell-1} - 2^{2\ell-7}) = 2^{-3} \cdot (1 - 2^{-6}).$$

By lemma 8 the classical algorithm in section 5.2 will upon input of a pair (j,k) for which j is good output d and so the theorem follows. ■

Again, it should be stressed that we have have not sought the best lower bound on the success probability in this paper.

5.4 Generalizations and observations

We remark that we have not used that q is a prime in the algorithm. Hence, the algorithm generalizes to finite abelian groups.

Furthermore, we remark that we have in fact not assumed q to be explicitly known in the algorithm. We have only assumed that $q \geq 2^{2\ell} + (2^{\ell} - 1)d$.

5.4.1 On whether q has to be explicitly known

The above observation not withstanding, q may need to be explicitly known for reasons that are not immediately apparent from the algorithm description.

For instance, to efficiently implement the exponentiations, the inverses $[-1]\,x$ and $[-1]\,g$ may have to be pre-computed classically and $[-1]\,x = [q-1]\,x$ for any finite abelian group of order q. However, there may be other ways to compute inverses, depending on the group. In the special case of subgroups of order q to $\mathbb{F}_p^*$, for example, knowing p is sufficient since $[-1]\,x = [q-1]\,x = [(p-1)-1]\,x$.

6 Summary and conclusion

We have revisited Shor's algorithm for computing discrete logarithms in $\mathbb{F}_p^*$ on a quantum computer and modified it to compute logarithms d in groups of prime order q in the special case where $d \lll q$. More specifically, we have considered the special case where $0 < d < 2^{\ell}$ and $q \geq 2^{2\ell} + (2^{\ell} - 1)d$. As a stepping stone to performing this modification, we first introduced a modified algorithm for computing logarithms on the general interval $0 < d < q$ for comparison.

We have demonstrated conservative lower bounds on the success probability in both the general and special cases using simple proof techniques.

In the special case where $d \lll q$, our algorithm uses $3\,\lceil \log_2 d \rceil$ qubits for the two index registers and requires the computation of two QFTs of size $2^{\lceil \log_2 d \rceil}$ and $2^{2\,\lceil \log_2 d \rceil}$ respectively, compared to $2\,\lfloor \log_2 q \rfloor$ qubits for the index registers and two QFTs both of size $2^{\lfloor \log_2 q \rfloor}$ in the general case.

A quantum circuit capable of computing $[e]\,g = [a]\,g \odot [-b]\,x$ is furthermore required, where $x = [d]\,g$, and where $0 \leq a < 2^{2\,\lceil \log_2 d \rceil}$ and $0 \leq b < 2^{\lceil \log_2 d \rceil}$ in the special case, compared to $0 \leq a, b < 2^{\lfloor \log_2 q \rfloor}$ in the general case. In

both cases, the choice of group operation and element representation affects the complexity of computing and storing $[e]\,g$.

These results imply that the complexity of computing discrete logarithms in prime order groups on a quantum computer can be made to depend not only on the choice of group, and on its order q, but also on the logarithm d.

In the special case where $d \lll q$, our algorithm does not require q to be prime, so this result generalizes to finite abelian group. Furthermore, as per the description of the algorithm, q does not need to be explicitly known.

The reduction in size of the first two registers is in itself not very significant since many additional qubits will typically be required to implement the two exponentiation operations. However, the fact that the exponents that enter into these operations are now short, and that the QFTs are of smaller sizes, reduces the complexity in the special case where $d \lll q$.

Remarks

The quantum algorithms in this paper are described in purely mathematical terms. It is assumed that some quantum registers are first initialized, that a quantum circuit is then executed and that the system is then finally observed in a measurement. The extent to which the specialized algorithm for the case where $d \lll q$ provides an advantage over the general algorithm depends on how this mathematical description is translated into a physical implementation.

Note furthermore that in the case of subgroups of prime order q to $\mathbb{F}_p^*$, for fixed p and $d \lll p$ the general algorithm has lower complexity when $d \sim q \lll p$ than does the specialized algorithm when $d \lll q \sim p$. It is possible that the specialized algorithm may be further optimized in this respect.

Acknowledgments

Many thanks to Johan Håstad for providing key comments, suggestions and guidance. A note of thanks also goes to Lennart Brynielsson for correcting several minor errors in early draft versions of this paper.

Funding and support for this work was provided by the Swedish NCSA that is a part of the Swedish Armed Forces.

References

[1] D. Boneh, R. J. Lipton, *"Quantum Cryptanalysis of Hidden Linear Functions"*, in proceedings from the International Cryptology Conference, Advances in Cryptology – CRYPTO '95, volume 963, 1995, pp. 424-437.

[2] D. Gordon, *"Discrete logarithms in GF(p) using the Number Field Sieve"*, in SIAM Journal on Discrete Mathematics, volume 6, 1993, pp. 124-138.

[3] M. Hirvensalo, *"Quantum Computing"*, 2nd edition, Natural Computing Series, Springer Verlag, 2004.

[4] R. Josza, *"Quantum Factoring, Discrete Logarithms, and the Hidden Subgroup Problem"*, in Computing in Science and Engineering, volume 3, no 2, 2001, pp. 34-43.

[5] A. K. Lenstra, H. W. Lenstra, L. Lovász, *"Factoring polynomials with rational coefficients"*, in Mathematische Annalen, volume 261, no 4, 1982, p.p. 515-534.

[6] P. C. van Oorschot, M. J. Wiener, *"On Diffie-Hellman Key Agreement with Short Exponents"*, in proceedings from the International Conference on the Theory and Application of Cryptographic Techniques, Advances in Cryptology – EUROCRYPT '96, volume 1070, 1996, pp. 332-343.

[7] S. C. Pohlig, M. E. Hellman, *"An Improved Algorithm for Computing Logarithms over GF(p) and Its Cryptographic Significance"*, in IEEE Transactions on Information Theory, volume IT-24, no 1, 1978, pp. 106-110.

[8] J. M. Pollard, *"Monte Carlo Methods for Index Computation (mod p)"*, in Mathematics of Computation, volume 32, no 143, 1978, pp. 918-924.

[9] O. Schirokauer, *"Discrete Logarithms and Local Units"*, in Philosophical Transactions of the Royal Society of London, volume A 345, 1993, pp. 409-423.

[10] P. W. Shor, *"Algorithms for Quantum Computation: Discrete Logarithms and Factoring"*, in proceeding from the 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM, November 20–22, 1994, IEEE Computer Society Press, pp. 124–134.

[11] P. W. Shor, *"Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer"*, in SIAM Journal of Computing, volume 26, no 5, 1997, pp. 1484-1509.