Quantum Algorithms for Computing Short Discrete Logarithms and Factoring RSA Integers

Martin Ekerå[1,2(✉)] and Johan Håstad[1]

[1] KTH Royal Institute of Technology, 100 44 Stockholm, Sweden
{ekera,johanh}@kth.se
[2] Swedish NCSA, Swedish Armed Forces, 107 85 Stockholm, Sweden

Abstract. We generalize the quantum algorithm for computing short discrete logarithms previously introduced by Ekerå [2] so as to allow for various tradeoffs between the number of times that the algorithm need be executed on the one hand, and the complexity of the algorithm and the requirements it imposes on the quantum computer on the other hand. Furthermore, we describe applications of algorithms for computing short discrete logarithms. In particular, we show how other important problems such as those of factoring RSA integers and of finding the order of groups under side information may be recast as short discrete logarithm problems. This gives rise to an algorithm for factoring RSA integers that is less complex than Shor's general factoring algorithm in the sense that it imposes smaller requirements on the quantum computer. In both our algorithm and Shor's algorithm, the main hurdle is to compute a modular exponentiation in superposition. When factoring an n bit integer, the exponent is of length $2n$ bits in Shor's algorithm, compared to slightly more than $n/2$ bits in our algorithm.

Keywords: Discrete logarithms · Factoring · RSA · Shor's algorithms

1 Introduction

In a groundbreaking paper [8] from 1994, subsequently extended and revised in a later publication [9], 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 cyclic group, provided the group operation may be implemented efficiently using quantum circuits.

1.1 Recent Work

Ekerå [2] has introduced a modified version of Shor's algorithm for computing short discrete logarithms in finite cyclic groups.

T. Lange and T. Takagi (Eds.): PQCrypto 2017, LNCS 10346, pp. 347–363, 2017.
DOI: 10.1007/978-3-319-59879-6_20

Unlike Shor's original algorithm, this modified algorithm does not require the order of the group to be known. It only requires the logarithm to be short; i.e. it requires the logarithm to be small in relation to the group order.

The modified algorithm is less complex than Shor's general algorithm when the logarithm is short. This is because the main hurdle in both algorithms is to compute a modular exponentiation in superposition.

In the case where the group order is of length l bits and the logarithm sought is of length $m \lll l$ bits, Ekerå's algorithm exponentiates two elements to exponents of size $2m$ bits and m bits respectively. In Shor's algorithm, both exponents are instead of size $l \ggg m$ bits.

This difference is important since it is seemingly hard to build and operate large and complex quantum computers. If the complexity of a quantum algorithm may be reduced, in terms of the requirements that it imposes on the quantum computer, this could potentially mean the difference between being able to execute the algorithm and not being able to execute the algorithm.

1.2 Our Contributions

In the first part of this paper, we generalize the algorithm of Ekerå by considering the setting where the quantum algorithm is executed multiple times to yield multiple partial results. We then combine these partial results using lattice-based techniques in a classical post-processing stage to yield the discrete logarithm.

This enables us to further reduce the size of the exponent to only slightly more than m bits. Furthermore, this enables us to make tradeoffs between the number of times that the quantum algorithm need be executed on the one hand, and the complexity of the algorithm and the requirements that it imposes on the quantum computer on the other hand.

In the second part of this paper, we describe applications for our algorithm for computing short discrete logarithms. In particular, we show how the problems of factoring RSA integers and of finding the order of groups under side information may be recast as short discrete logarithm problems. By RSA integer we mean an integer that is the product of two primes of similar size.

This immediately gives rise to an algorithm for factoring RSA integers that is less complex than Shor's original general factoring algorithm in terms of the requirements that it imposes on the quantum computer. When factoring an n bit integer using Shor's order finding algorithm, an exponentiation is performed to an exponent of length $2n$ bits. In our algorithm, the corresponding exponent is instead of length $(\frac{1}{2} + \frac{1}{s})n$ bits where $s \geq 1$ is a parameter that may assume any small integer value. The exponent length is hence reduced from $2n$ bits to slightly more than $n/2$ bits.

1.3 Related Work

In an earlier work, unknown to us at the time of developing these results, Seifert [7] proposed to execute Shor's order finding algorithm multiple times with a

smaller range of exponents to obtain a set of partial results, and to compute the order from this set using simultaneous Diophantine approximation techniques. He gives an algorithm that uses an exponent of length slightly more than n bits when factoring an n bit RSA integer.

Seifert does not give all details in his analysis of the algorithm but several of his ideas parallel those of ours. In this work, however, we further reduce the length of the exponent to slightly more than $n/2$ bits by casting the RSA factoring problem as a short discrete logarithm problem and solving it using our generalized algorithm. The fact that we do not have any modular reductions in the exponent makes for a simpler and more transparent proof.

1.4 Overview

In the next section, we introduce notation that is used throughout the paper, and in Sect. 3 we describe our generalized algorithm for computing short discrete logarithms, briefly discuss its applications, and describe its advantage.

We proceed to describe algorithms for factoring RSA integers in Sect. 4 and for finding the order of group elements under side information in Sect. 5. Both of these algorithms are based upon the generalized algorithm for computing short discrete logarithms introduced in Sect. 3.

2 Notation

In this section, we introduce some notation used throughout this paper.

- $u \bmod n$ denotes u reduced modulo n and constrained to $0 \leq u \bmod n < n$.
- $\{u\}_n$ denotes u reduced modulo n and constrained to $-n/2 \leq \{u\}_n < n/2$.
- $\lfloor u \rceil$ denotes u rounded to the closest integer.
- $|\, a + ib \,| = \sqrt{a^2 + b^2}$ where $a, b \in \mathbb{R}$ denotes the Euclidean norm of $a + ib$.
- $|\, \boldsymbol{u} \,|$ denotes the Euclidean norm of the vector $\boldsymbol{u} = (u_0, \ldots, u_{n-1}) \in \mathbb{R}^n$.

3 Computing Short Discrete Logarithms

In this section, we describe a generalization of the algorithm for computing short discrete logarithms previously introduced by Ekerå [2].

3.1 The Discrete Logarithm Problem

Let $\mathbb{G}$ under $\odot$ be a group of order r 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 $d = \log_g x$.

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.

3.2 Algorithm Overview

The generalized algorithm for computing short discrete logarithms consists of two stages; an initial quantum stage and a classical post-processing stage.

The initial quantum stage is described in terms of a quantum algorithm, see Sect. 3.3, that upon input of g and $x = [d]\,g$ yields a pair (j, k). The classical post-processing stage is described in terms of a classical algorithm, see Sect. 3.9, that upon input of $s \geq 1$ "good" pairs computes and returns d.

The parameter s determines the number of good pairs (j, k) required to successfully compute d. It furthermore controls the sizes of the index registers in the algorithm, and thereby the complexity of executing the algorithm on a quantum computer and the sizes of and amount of information on d contained in the two components j, k of each pair.

In the special case where $s = 1$ the generalized algorithm is identical to the algorithm in [2]. A single good pair then suffices to compute d.

By allowing s to be increased, the generalized algorithm enables a tradeoff to be made between the requirements imposed by the algorithm on the quantum computer on the one hand, and the number of times it needs to be executed and the complexity of the classical post-processing stage on the other hand.

We think of s as a small constant. Thus, when we analyze the complexity of the algorithm, and in particular the parts of the algorithm that are executed classically, we can neglect constants that depend on s.

3.3 The Quantum Algorithm

Let m be the smallest integer such that $0 < d < 2^m$ and let ℓ be an integer close to m/s. Provided that the order r of g is at least $2^{\ell+m} + 2^\ell d$, the quantum algorithm described in this section will upon input of g and $x = [d]\,g$ compute and output a pair (j, k).

A set of such pairs is then input to the classical algorithm to recover d.

1. Let
$$|\,\Psi\,\rangle = \frac{1}{\sqrt{2^{2\ell+m}}} \sum_{a=0}^{2^{\ell+m}-1} \sum_{b=0}^{2^\ell-1} |\,a\,\rangle\,|\,b\,\rangle\,|\,0\,\rangle\,.$$
2. Compute $[a]\,g \odot [-b]\,x$ and store the result in the third register
$$|\,\Psi\,\rangle = \frac{1}{\sqrt{2^{2\ell+m}}} \sum_{a=0}^{2^{\ell+m}-1} \sum_{b=0}^{2^\ell-1} |\,a, b, [a - bd]\,g\,\rangle\,.$$
3. Compute a QFT of size $2^{\ell+m}$ of the first register and a QFT of size 2^ℓ of the second register to obtain
$$|\,\Psi\,\rangle = \frac{1}{2^{2\ell+m}} \sum_{a,\,j=0}^{2^{\ell+m}-1} \sum_{b,\,k=0}^{2^\ell-1} \mathrm{e}^{2\pi i\,(aj+2^m bk)/2^{\ell+m}}\,|\,j, k, [a\,-\,bd]\,g\,\rangle\,.$$
4. Observe the system in a measurement to obtain (j, k) and $[e]\,g$.

3.4 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

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

where the sum is over all pairs (a, b) that produce this specific e. Note that the assumption that the order $r \geq 2^{\ell+m} + 2^\ell d$ implies that no reduction modulo r occurs when e is computed.

Since $e = a - bd$ we have $a = e + bd$. Substituting a for this expression in (1), extracting the term containing e, and centering b around zero yields

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

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

Note that a modular reduction by $2^{\ell+m}$ has furthermore been introduced. This is possible since adding or subtracting multiples of $2^{\ell+m}$ has no effect; it is equivalent to shifting the phase angle by a multiple of 2π.

3.5 The Notion of Constructive Interference

The claim below summarizes the notion of constructive interference that we use to lower-bound the success probability.

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 that $\frac{1}{\sqrt{2}} \leq \cos\theta_j \leq 1$ and so the claim follows. □

3.6 The Notion of a Good Pair (j, k)

By Claim 1 the sum in Eq. (2) is large when $|\, \{dj + 2^m k\}_{2^{\ell+m}} \,| \leq 2^{m-2}$ 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 any specific good pair.

Definition 1. *A pair (j, k), where j and k are integers such that $0 \leq j < 2^{\ell+m}$ and $0 \leq k < 2^{\ell}$, is said to be* good *if $|\{dj + 2^m k\}_{2^{\ell+m}}| \leq 2^{m-2}$.*

Note that j uniquely defines k as k gives the ℓ high order bits of dj modulo $2^{\ell+m}$ or more specifically as $k = -\lfloor (dj \bmod 2^{\ell+m})/2^m \rceil \bmod 2^{\ell}$.

3.7 Lower-Bounding the Number of Good Pairs (j, k)

Lemma 1. *There are at least $2^{\ell+m-1}$ different j such that there is a k such that (j, k) is a good pair.*

Proof. For a good pair

$$|\{dj + 2^m k\}_{2^{\ell+m}}| = |\{dj\}_{2^m}| \leq 2^{m-2} \tag{3}$$

and for each j that satisfies (3) there is a unique k such that (j, k) is good.

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

Assume that $\kappa = m - 1$. Then the only possible values are 0 and 2^{m-1}. Only zero gives rise to a good pair. With multiplicity there are $2^{\ell+\kappa} = 2^{\ell+m-1}$ integers j such that (j, k) is a good pair. Assume that $\kappa < m - 1$. Then only the $2 \cdot 2^{m-\kappa-2} + 1$ values congruent to values on $[-2^{m-2}, 2^{m-2}]$ are such that $|\{dj\}_{2^m}| \leq 2^{m-2}$. With multiplicity $2^{\ell+\kappa}$ there are $2^{\ell+\kappa} \cdot (2 \cdot 2^{m-\kappa-2} + 1) \geq 2^{\ell+m-1}$ integers j such that (j, k) is a good pair. In both cases there are at least $2^{\ell+m-1}$ good pairs and so the lemma follows. □

3.8 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 $e = a - bd$ where a, b are integers on the intervals $0 \leq a < 2^{\ell+m}$ and $0 \leq b < 2^{\ell}$.*

Claim 2.

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

Proof. Since a, b may independently assume $2^{\ell+m}$ and 2^{ℓ} values, there are $2^{2\ell+m}$ distinct pairs (a, b). From this fact, and from the fact that $|e = a - bd| < 2^{\ell+m}$ since $0 \leq a < 2^{\ell+m}$, $0 \leq b < 2^{\ell}$ and $0 < d < 2^m$, the claim follows. □

Claim 3.

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

Proof. *This follows from the Cauchy–Schwarz inequality and Claim 2 since*

$$2^{2(2\ell+m)} = \left(\sum_{e=-2^{\ell+m}}^{2^{\ell+m}-1} T_e \right)^2 \leq \left(\sum_{e=-2^{\ell+m}}^{2^{\ell+m}-1} 1^2 \right) \left(\sum_{e=-2^{\ell+m}}^{2^{\ell+m}-1} T_e^2 \right).$$

□

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

Lemma 2. *The probability of obtaining any specific good pair* (j, k) *from a single execution of the algorithm in Sect. 3.3 is at least* $2^{-m-\ell-2}$.

Proof. For a good pair

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

for any integer b on the interval $0 \leq b < 2^\ell$. It therefore follows from Claim 1 that the probability of observing (j, k) and $[e]\, g$ is at least

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

Summing this over all e and using Claim 3 yields

$$\sum_{e=-2^{\ell+m}}^{2^{\ell+m}-1} \frac{T_e^2}{2 \cdot 2^{2(2\ell+m)}} \geq 2^{-m-\ell-2}$$

from which the lemma follows. □

We note that by Lemmas 1 and 2 the probability of the algorithm yielding a good pair as a result of a single execution is at least 2^{-3}.

3.9 Computing d from a Set of s Good Pairs

In this section, we specify a classical algorithm that upon input of a set of s distinct good pairs $\{(j_1, k_1), \ldots, (j_s, k_s)\}$, that result from multiple executions of the algorithm in Sect. 3.3, computes and outputs d.

Definition 3. *Let* L *be the integer lattice generated by the row span of*

$$\begin{bmatrix} j_1 & j_2 & \cdots & j_s & 1 \\ 2^{\ell+m} & 0 & \cdots & 0 & 0 \\ 0 & 2^{\ell+m} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 2^{\ell+m} & 0 \end{bmatrix}.$$

The algorithm proceeds as follows to recover d from $\{(j_1, k_1), \ldots, (j_s, k_s)\}$.

1. Let $\boldsymbol{v} = (\{-2^m k_1\}_{2^{\ell+m}}, \ldots, \{-2^m k_s\}_{2^{\ell+m}}, 0) \in \mathbb{Z}^{s+1}$.
   For all vectors $\boldsymbol{u} \in L$ such that
   $$|\boldsymbol{u} - \boldsymbol{v}| < \sqrt{s/2^4 + 1} \cdot 2^m$$
   test if the last component of $\boldsymbol{u}$ is d by checking if $x = [d]\, g$. If so return d.
2. If d is not found in step 1 or the search is infeasible the algorithm fails.

As s is a small constant, all vectors close to $\boldsymbol{v}$ can be explored efficiently.

The most straightforward way to explore all vectors close to $\boldsymbol{v}$ is to first compute a reduced basis for L using standard basis reduction techniques such as Lenstra-Lenstra-Lovász (LLL) [4]. The reduced basis may then be used to find the vector in L closest to $\boldsymbol{v}$, and to explore the neighborhood of this vector in L by adding and subtracting vectors from the basis. There are however more efficient ways to enumerate the vectors in lattices, see for instance the work of Micciancio and Walter [5].

One problem that remains to be addressed is that there may theoretically exist many vectors in L that are close to $\boldsymbol{v}$. In Lemma 3 in the next section we demonstrate that this is not the case with high probability.

3.10 Rationale and Analysis

For any $m_1, \ldots, m_s \in \mathbb{Z}$ the vector

$$\boldsymbol{u} = (\{dj_1\}_{2^{\ell+m}} + m_1 2^{\ell+m}, \ldots, \{dj_s\}_{2^{\ell+m}} + m_s 2^{\ell+m}, d) \in L.$$

The above algorithm performs an exhaustive search of all vectors in L at distance at most $\sqrt{s/2^4 + 1} \cdot 2^m$ from $\boldsymbol{v}$ to find $\boldsymbol{u}$ for some $m_1, \ldots, m_s$. It then recovers d as the second component of $\boldsymbol{u}$. The search will succeed in finding $\boldsymbol{u}$ since

$$\begin{aligned} |\boldsymbol{u} - \boldsymbol{v}| &= \sqrt{d^2 + \sum_{i=1}^{s} \left(\{dj_i\}_{2^{\ell+m}} + m_i\, 2^{\ell+m} - \{-2^m k_i\}_{2^{\ell+m}}\right)^2} \\ &= \sqrt{d^2 + \sum_{i=1}^{s} \left(\{dj_i + 2^m k_i\}_{2^{\ell+m}}\right)^2} < \sqrt{s/2^4 + 1} \cdot 2^m \end{aligned}$$

since $0 < d < 2^m$ and $|\{dj + 2^m k\}_{2^{\ell+m}}| \leq 2^{m-2}$ by the definition of a good pair, and since $m_1, \ldots, m_s$ may be freely selected to obtain equality.

Whether the search is computationally feasible depends on the number of vectors in L that lie within distance $\sqrt{s/2^4 + 1} \cdot 2^m$ of $\boldsymbol{v}$. This number is related to the norm of the shortest vector in the lattice.

Note that the determinant of L is $2^{(\ell+m)s} \approx 2^{m(s+1)}$. As the lattice is $s+1$-dimensional we would expect the shortest vector to be of length about 2^m. This is indeed true with high probability.

Lemma 3. *The probability that L contains a vector $\boldsymbol{u} = (u_1, \ldots u_{s+1})$ with $|\, u_i \,| < 2^{m-3}$ for $1 \leq i \leq s+1$ is bounded by 2^{-s-1}.*

Proof. Take any integer $\boldsymbol{u}$ with all coordinates strictly bounded by 2^{m-3}.

If 2^κ is the largest power of two that divides u_{s+1} then u_i must also be divisible by 2^κ for $\boldsymbol{u}$ to belong to any lattice in the family. By family we mean all lattices on the same form and degree as L, see Definition 3. If it this is true for all i then $\boldsymbol{u}$ belongs to L for $2^{s\kappa}$ different values of $(j_i)_{i=1}^s$.

There are $2^{(m-2-\kappa)(s+1)}$ vectors $\boldsymbol{u}$ with all coordinates divisible by 2^κ and bounded in absolute value by 2^{m-3}. We conclude that the total number of lattices L that contain such a short vector is bounded by

$$\sum_\kappa 2^{(m-2-\kappa)(s+1)} \cdot 2^{\kappa s} \leq 2^{1+(m-2)(s+1)}.$$

As the number of s-tuples of good j is at least $2^{s(\ell+m-1)}$, the lemma follows. □

Lemma 3 shows that with good probability the number of lattice points such that $|\, \boldsymbol{u} - \boldsymbol{v} \,| < \sqrt{s/2^4 + 1} \cdot 2^m$ is a constant that only depends on s and thus we can efficiently find all such vectors.

3.11 Building a Set of s Good Pairs

The probability of a single execution of the quantum algorithm in Sect. 3.3 yielding a good pair is at least 2^{-3} by Lemmas 1 and 2. Hence, if we execute the quantum algorithm $t = 8s$ times, we obtain a set of t pairs that we expect contains at least s good pairs.

In theory, we may then recover d by executing the classical algorithm in Sect. 3.9 with respect to all $\binom{t}{s}$ subsets of s pairs selected from this set. Since s is a constant, this approach implies a constant factor overhead in the classical part of the algorithm. It does not affect the quantum part of the algorithm. We summarize these ideas in Theorem 4 below.

In practice, however, we suspect that it may be easier to recover d. First of all, we remark that we have only established a lower bound on the probability that a good pair is yielded by the algorithm. This bound is not tight and we expect the actual probability to be higher than is indicated by the bound.

Secondly, we have only analyzed the probability of the classical algorithm in Sect. 3.9 recovering d under the assumption that all s pairs in the set input are good. It might however well turn out to be true that the algorithm will succeed in recovering d even if not all pairs in the input set are good.

3.12 Main Result

In this subsection we summarize the above discussion in a main theorem. Again, we stress that the approach outlined in the theorem is conservative.

Theorem 4. *Let d be an integer on $0 < d < 2^m$, let $s \geq 1$ be a fixed integer, let ℓ be an integer close to m/s and let g be a generator of a finite cyclic group of order $r \geq 2^{\ell+m} + 2^{\ell} d$. Then there exists a quantum algorithm that yields a pair as output when executed with g and $x = [d]\, g$ as input. The main operation in this algorithm is an exponentiation of g in superposition to an exponent of length $\ell + m$ bits. If this algorithm is executed cs times for some small constant c to yield a set of pairs $\mathcal{S}$, then there exists a polynomial time classical algorithm that computes d if executed with all unordered subsets of s pairs from $\mathcal{S}$ as input.*

The proof of the quantum part of Theorem 4 follows from the above discussion.

Again, we remind the reader that s is a small constant. The complexity of the algorithm would otherwise be exponential in s. Furthermore, we note that the order r of the group need not be explicitly known. It suffices that the above requirement on r is met. Note furthermore that it must be possible to implement the group operation efficiently on a quantum computer.

3.13 The Advantage of Our Algorithm

The advantage of using our algorithm to compute short discrete logarithms in comparison to using Shor's general algorithm is that our algorithm imposes smaller requirements on the quantum computer.

To explain in which respects our algorithm imposes smaller requirements on the quantum computer, we first note that in both our algorithm and Shor's algorithm the quantum stages consist of two modular exponentiations followed by the computation of two QFTs, see Figs. 1 and 2 in the appendix.

Except for the reduction in the exponent lengths and in the sizes of the QFTs in our algorithm compared to Shor's algorithm, the quantum stages of the two algorithms are identical. The advantage provided by our algorithm over Shor's algorithm when the logarithm is short stems from these reductions.

Second, we note that a QFT may be computed efficiently when the size is a power of two as described by Shor [8,9]. In fact, the QFTs may be computed on-the-fly using only a single control qubit as we will elaborate on in further detail below. Hence, the main hurdle is the modular exponentiation. For a more in-depth discussion, see for instance the work by Cleve and Watrous [1].

The standard quantum algorithm for exponentiating a fixed group element as described by Shor [9] is to classically pre-compute consecutive powers of the two elements, and to construct quantum circuits for combining these pre-computed powers with generic group elements under the group operation. These circuits are then executed in sequence controlled by one or more qubits in a separate index register. The classical counterpart to this algorithm is the square-and-multiply algorithm, or the double-and-add algorithm, if the group is written additively.

Assuming the standard algorithm is used to implement the exponentiation, the number of such group operations that need be implemented in the quantum circuit is reduced by a constant factor directly proportional to the reduction in the exponent length. The reduction in the number of group operations that

need be implemented in the circuit implies a corresponding reduction both in the circuit depth and in the time required to execute the circuit, which translates into a reduction in the time that the quantum system need be kept coherent.

As for the register of qubits used to control the group operations, its size is reduced by a constant factor in implementations where a distinct qubit is used to control each group operation. However, it is possible to use a single control qubit to control all group operations, as described by Mosca and Ekert [6]. In such implementations, the number of qubits required to implement our algorithm compared to Shor's algorithm is not reduced.

As is evident from the above discussion, the advantage of using our algorithm to compute short discrete logarithms compared to using Shor's general algorithm depends on how the algorithms are implemented, but it always stems from the reduction in the length of the exponents and in the sizes of the QFTs.

3.14 Implementation Remarks

It should be explicitly stated that we describe our algorithm mathematically in this paper in terms of it using two index registers, and in terms of the quantum system being initialized, of a circuit then being executed, and of the system then being observed in a measurement.

This algorithm description, whilst being useful when seeking to understand and analyze the algorithm, is not necessarily representative of the manner in which the algorithm would be implemented in practice. To base comparisons solely upon this description without taking implementation details into account may prove misleading. This was demonstrated in the previous section.

Furthermore, it should be stated that there are various papers on how Shor's algorithm may be implemented in the literature. Many of these ideas carry over to our algorithm since the quantum stages are very similar.

3.15 Applications

Quantum algorithms for computing short discrete logarithms have cryptanalytic applications with respect to cryptanalysis of schemes in which the security relies on the intractability of the short discrete logarithm problem.

A concrete example of such an application is to attack Diffie-Hellman over finite fields when safe prime groups are used in conjunction with short exponents.

The existence of efficient specialized algorithms for computing short discrete logarithms on quantum computers should be taken into account when selecting and comparing domain parameters for asymmetric cryptographic schemes that rely on the computational intractability of the discrete logarithm problem.

For further details, the reader is referred to the extended rationale in [2] and to the references to the literature provided in that paper.

4 Factoring RSA Integers

In this section we describe how the RSA integer factoring problem may be recast as a short discrete logarithm problem by using ideas from Håstad et al. [3], and the fact that our algorithm does not require the group order to be known.

This immediately gives rise to an algorithm for factoring RSA integers that imposes smaller requirements on the quantum computer than Shor's general factoring algorithm.

4.1 The RSA Integer Factoring Problem

Let $p, q \neq p$ be two random odd primes such that $2^{n-1} < p, q < 2^n$. The RSA integer factoring problem is then to factor $N = pq$ into p and q.

4.2 The Factoring Algorithm

Consider the multiplicative group $\mathbb{Z}_N^*$ to the ring of integers modulo N. This group has order $\phi(N) = (p-1)(q-1)$. Let $\mathbb{G}$ be some cyclic subgroup to $\mathbb{Z}_N^*$.

Then $\mathbb{G}$ has order $\phi(N)/t$ for some $t \mid \phi(N)$ such that $t \geq \gcd(p-1, q-1)$. In what follows below, we assume that $\phi(N)/t > (p+q-2)/2$.

1. Let g be a generator of $\mathbb{G}$. Compute $x = g^{(N-1)/2}$. Then $x \equiv g^{(p+q-2)/2}$.
2. Compute the short discrete logarithm $d = (p+q-2)/2$ from g and x.
3. Compute p and q by solving the quadratic equation

$$N = (2d - q + 2)q = 2(d+1)q - q^2$$

where we use that $2d + 2 = p + q$. This yields

$$p, q = c \pm \sqrt{c^2 - N} \quad \text{where} \quad c = d + 1.$$

We obtain p or q depending on the choice of sign.

To understand why we obtain a short logarithm, note that

$$N - 1 = pq - 1 = (p-1) + (q-1) + (p-1)(q-1)$$

from which it follows that $(N-1)/2 \equiv (p+q-2)/2 \bmod \phi(N)/t$ provided that the above assumption that $\phi(N)/t > (p+q-2)/2$ is met.

The only remaining difficulties are the selection of the generator in step 1 and the computation of the short discrete logarithm in step 2.

In step 1 we may pick any cyclic subgroup $\mathbb{G}$ to $\mathbb{Z}_N^*$ for as long as its order $\phi(N)/t$ is sufficiently large. It suffices that $\phi(N)/t > (p+q-2)/2$ and that the discrete logarithm can be computed, see Sect. 4.2 below for more information.

This implies that we may simply select an element g uniformly at random on the interval $1 < g < N-1$ and use it as the generator.

To compute the short discrete logarithm in step 2, we use the algorithm in Sect. 3. This algorithm requires that the order

$$\phi(N)/t \geq 2^{\ell+m} + 2^{\ell} d \quad \Rightarrow \quad \phi(N)/t \geq 2^{\ell+m+1}$$

where we have used that $0 < d < 2^m$. We note that

$$2^n \leq d = (p+q-2)/2 < 2^{n+1} \quad \Rightarrow \quad m = n+1.$$

Furthermore, we note that $\phi(N) = (p-1)(q-1) \geq 2^{2(n-1)}$ which implies

$$\phi(N)/t \geq 2^{2(n-1)}/t \geq 2^{\ell+m+1} = 2^{\ell+n+2} \quad \Rightarrow \quad t < 2^{2(n-1)-(n+\ell+2)} = 2^{n-\ell-4}.$$

Recall that $\ell = m/s = (n+1)/s$ where $s \geq 1$. For random p and q, and a randomly selected cyclic subgroup to $\mathbb{Z}_N^*$, the requirement $t < 2^{n-\ell-4}$ is hence met with overwhelming probability for any $s > 1$.

We remark that further optimizations are possible. For instance the size of the logarithm may be reduced by computing $x = g^{(N-1)/2-2^n}$ since $p, q > 2^{n-1}$.

4.3 Generalizations

We note that the algorithm proposed in this section can be generalized.

In particular, we note that we need not assume that the two factors are of the same length in bits. It suffices that the difference in length between the two factors is not too great for a short discrete logarithm problem to arise.

4.4 The Advantage of Our Algorithm

The advantage of using our algorithm to factor RSA integers in comparison to using Shor's general integer factoring algorithm is that our algorithm imposes smaller requirements on the quantum computer.

To compute two exponentiations with respect to different base elements and two QFTs, as in the discrete logarithm algorithms previously described, is not significantly different from computing a single larger exponentiation followed by a single QFT of large size, as in Shor's order finding algorithm that is the quantum part of Shor's factoring algorithm. For more information, see [8,9] and the quantum circuits in Figs. 1, 2 and 3 in the appendix.

In analogy with the situation in Sect. 3.13, the advantage of our algorithm over Shor's algorithm is again obtained by the total exponent length and the total size of the QFT being reduced by a constant factor.

5 Order Finding Under Side Information

In this section, we briefly consider the problem of computing the order of a cyclic group $\mathbb{G}$ when a generator g for the group is available and when side information is available in the form of an estimate of the group order.

Let $\mathbb{G}$ be a cyclic group of order r. Let r_0 be a known approximation of the order such that $0 \leq r - r_0 < 2^m$. The problem of computing the order r under the side information r_0 may then be recast as a short discrete logarithm problem:

1. Let g be a generator of $\mathbb{G}$. Compute $x = g^{-r_0}$. Then $x \equiv g^{r-r_0}$.
2. Compute the short discrete logarithm $d = r - r_0$ from g and x.
3. Compute the order $r = d + r_0$.

6 Summary and Conclusion

We have generalized the quantum algorithm for computing short discrete logarithms previously introduced by Ekerå [2] so as to allow for various tradeoffs between the number of times that the algorithm need be executed on the one hand, and the complexity of the algorithm and the requirements it imposes on the quantum computer on the other hand.

In the case where the group order is of length l bits and the logarithm sought is of length $m \lll l$ bits, Ekerå's algorithm exponentiates two elements to exponents of size $2m$ bits and m bits respectively. In Shor's algorithm for computing discrete logarithms, both exponents are instead of size l bits. Our generalized algorithm reduces this to $\ell + m$ and ℓ bits where $\ell \approx m/s$ and $s \geq 1$ is a small integer constant. For $s = 1$ our algorithm is identical to Ekerå's algorithm.

These algorithms have immediate cryptanalytic applications with respect to cryptanalysis of schemes in which the security rests on the intractability of the short discrete logarithm problem. Furthermore, we have described additional applications for algorithms for computing short discrete logarithms.

In particular, we have shown how other important problems such as those of factoring RSA integers and of finding the order of groups under side information may be recast as short discrete logarithm problems. This immediately gives rise to an algorithm for factoring RSA integers that is less complex than Shor's general factoring algorithm in the sense that it imposes smaller requirements on the quantum computer.

In both our algorithm and Shor's algorithm, the main hurdle is to compute a modular exponentiation in superposition. When factoring an n bit integer, the exponent is of length $2n$ bits in Shor's algorithm, compared to slightly more than $n/2$ bits in our algorithm. We have made essentially two optimizations that give rise to this improvement:

First, we gain a factor of two by re-writing the factoring problem as a short discrete logarithm problem and solving it using our algorithm for computing short discrete logarithms. One way to see this is that we know an approximation N of the order $\phi(N)$. This gives us a short discrete logarithm problem and our algorithm for solving it does not require the order to be known beforehand.

Second, we gain a factor of two by executing the quantum algorithm multiple times to yield a set of partial results. We then recover the discrete logarithm d from this set in a classical post-processing step. The classical algorithm uses lattice-based techniques.

Acknowledgments. Support for this work was provided by the Swedish NCSA, that is a part of the Swedish Armed Forces, and by the Swedish Research Council (VR). We are grateful to Lennart Brynielsson for many interesting discussions on the topic of this paper. The input of the referees and of Rainer Steinwandt was also helpful.

A Appendix

In this appendix we provide graphical visualizations of some of the quantum circuits described earlier so as to facilitate the reader's comprehension.

All circuits below use the standard algorithm for exponentiation and one control qubit for each group operation as described in Sect. 3.13. Recall that a single qubit may be used to control all operations as described by Mosca and Ekert [6]. This reduces the topmost registers in the figures to a single qubit.

We assume below that t qubits is sufficient to represent group elements and to perform the required group operations. Furthermore, we introduce the controlled operator U_v that upon input of $|\, u \,\rangle$ where $u, v \in \mathbb{G}$ outputs $|\, u \odot v \,\rangle$ if the control qubit is $|\, 1 \,\rangle$ and $|\, u \,\rangle$ otherwise.

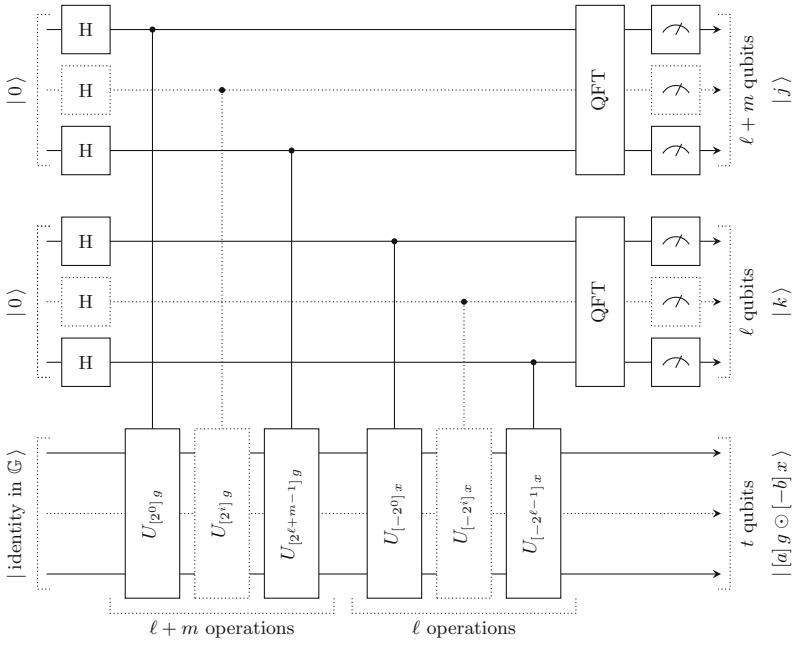

Fig. 1. A quantum circuit for the quantum stage in the algorithm for computing short discrete logarithms described in Sect. 3.

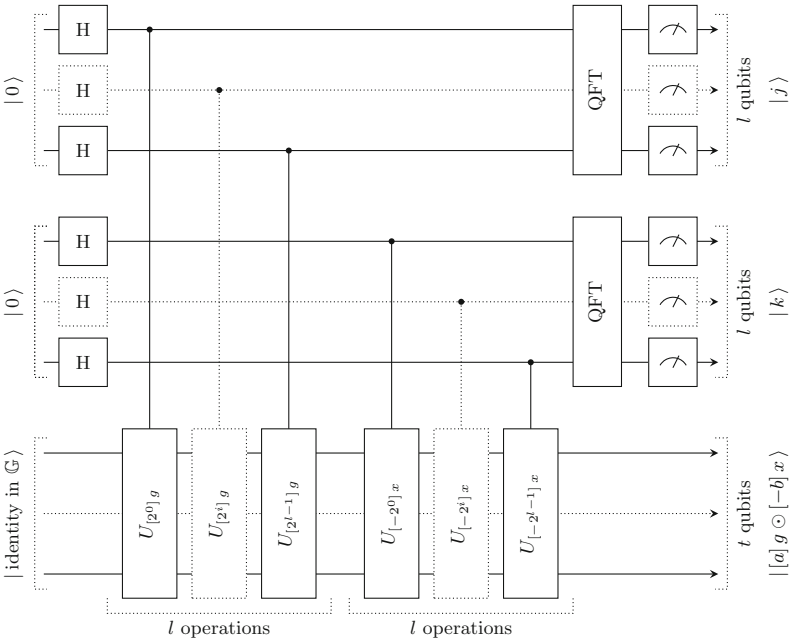

Fig. 2. A quantum circuit for the quantum stage in Shor's algorithm for computing general discrete logarithms [2,8,9]. It is identical to the circuit in Fig. 1, except that the exponent lengths and register sizes are larger. In this figure, l denotes the length in bits of the order of $\mathbb{G}$. The circuit in Fig. 1 has an advantage when $l \ggg \ell + m/2$.

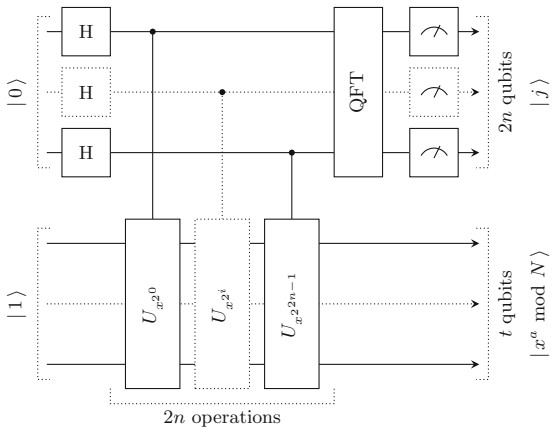

Fig. 3. A quantum circuit for the quantum stage in the order finding algorithm that is part of Shor's factoring algorithm [8,9]. In this figure, x is a random integer, n denotes the length in bits of the integer N to be factored and we assume $\mathbb{G} \subseteq \mathbb{Z}_N^*$.

References

1. Cleve, R., Watrous, J.: Fast parallel circuits for the quantum Fourier transform. In: Proceedings of the 41st Annual Symposium on Foundations of Computer Science, pp. 526–536 (2000)
2. Ekerå, M.: Modifying Shor's algorithm to compute short discrete logarithms. Cryptology ePrint Archive, Report 2016/1128 (2016)
3. Håstad, J., Schrift, A., Shamir, A.: The discrete logarithm modulo a composite hides O(n) bits. J. Comput. Syst. Sci. **47**(3), 376–404 (1993)
4. Lenstra, H.W., Lenstra, A.K., Lovász, L.: Factoring polynomials with rational coefficients. Math. Ann. **261**, 515–534 (1982)
5. Micciancio, D., Walter, M.: Fast lattice point enumeration with minimal overhead. In: Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 276–294 (2015)
6. Mosca, M., Ekert, A.: The hidden subgroup problem and eigenvalue estimation on a quantum computer. In: Williams, C.P. (ed.) QCQC 1998. LNCS, vol. 1509, pp. 174–188. Springer, Heidelberg (1999). doi:10.1007/3-540-49208-9_15
7. Seifert, J.-P.: Using fewer qubits in Shor's factorization algorithm via simultaneous diophantine approximation. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 319–327. Springer, Heidelberg (2001). doi:10.1007/3-540-45353-9_24
8. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings of the 35th Annual Symposium on Foundations of Computer Science, pp. 124–134 (1994)
9. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput. **26**(5), 1484–1509 (1997)