On post-processing in the quantum algorithm for computing short discrete logarithms

Martin Ekerå[1,2]

[1]KTH Royal Institute of Technology, Stockholm, Sweden
[2]Swedish NCSA, Swedish Armed Forces, Stockholm, Sweden

February 10, 2019

Abstract

We revisit the quantum algorithm for computing short discrete logarithms that was recently introduced by Ekerå and Håstad. By carefully analyzing the probability distribution induced by the algorithm, we show its success probability to be higher than previously reported. Inspired by our improved understanding of the distribution, we propose an improved post-processing algorithm that is practical, enables better tradeoffs to be achieved, and requires fewer runs, than the original post-processing algorithm. To prove these claims, we construct a classical simulator for the quantum algorithm by sampling the probability distribution it induces for given logarithms. This simulator is in itself a key contribution. We use it to demonstrate that our quantum algorithm achieves an advantage over Shor's algorithms, not only in each individual run, but also overall, when targeting cryptographically relevant instances of RSA and Diffie-Hellman with short exponents.

1 Introduction

In a groundbreaking paper [7] from 1994, subsequently extended and revised in a later publication [8], Shor introduced polynomial time quantum computer algorithms for factoring integers and for computing discrete logarithms in $\mathbb{F}_p^*$.

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.

More recently, Ekerå [2] introduced a modified version of Shor's algorithm for computing discrete logarithms that is more efficient than Shor's original algorithm when the logarithm is short. This work was originally motivated by the use of short discrete logarithms in instantiations of cryptographic schemes based on the computational intractability of the discrete logarithm problem in finite fields. A concrete example is the use of short exponents in the Diffie-Hellman key exchange protocol when instantiated with safe-prime groups.

This work was subsequently generalized by Ekerå and Håstad [3] so as to enable tradeoffs between the number of times that the algorithm needs to be executed, and the requirements it imposes on the quantum computer. These ideas parallel earlier ideas by Seifert [6] for making tradeoffs in Shor's order finding algorithm; the quantum part of Shor's general factoring algorithm.

Ekerå and Håstad furthermore explained how the RSA integer factoring problem may be expressed as a short discrete logarithm problem, giving rise to a new algorithm for factoring RSA integers that imposes less requirements on the quantum computer than Shor's general factoring algorithm taking into account Seifert's tradeoffs. The new algorithm does not directly rely on order finding.

As it is seemingly difficult to construct and operate large-scale quantum computers, any reduction in the requirements imposed on the computer by the algorithm when solving cryptographically relevant problems is potentially important and merits study. In this paper we improve the post-processing in the aforementioned algorithms to further reduce the requirements on the quantum computer when computing short discrete logarithms and factoring RSA integers.

1.1 Preliminaries

To compute an m bit logarithm d, the generalized algorithm by Ekerå and Håstad [3] exponentiates group elements in superposition to $m + 2m/s$ bit exponents for $s \geq 1$ an integer. Given a set of s good outputs, a classical post-processing algorithm then recovers d by enumerating an $s + 1$-dimensional lattice.

The number of runs required increases in s, in exchange for a reduction in the exponent length in the quantum algorithm. Hence, increasing s trades work in each quantum algorithm run for work in the classical post-processing. Assuming use of sequential double-and-add exponentiation, the quantum circuit size and depth, and the required coherence time, are thus all reduced by a constant factor.

To minimize the work performed in each run, s would ideally be selected large. However, as good outputs cannot be efficiently recognized, the algorithm needs to be run s/p times and all subsets of s outputs exhaustively post-processed, where p denotes the probability of obtaining a good output. This exponential complexity in s restricts the achievable tradeoff.

One of our key contributions in this paper is to show that if outputs from n runs of the quantum algorithm are included in an $n + 1$-dimensional lattice, for some s and sufficiently large $n > s$, then d may be recovered by reducing the lattice and applying Babai's nearest plane algorithm [1]. Enumerating exponential in s many $s + 1$-dimensional lattices may thus be avoided, at the expense of reducing a single $n+1$-dimensional lattice. The new post-processing algorithm is very efficient, practical for comparatively large s, and requires considerably fewer quantum algorithm runs, than the post-processing algorithm originally proposed.

1.2 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 $d = \log_g x$. In the general discrete logarithms problem $0 \leq d < r$, whereas d is smaller than r by some order of magnitude in the short discrete logarithm problem.

1.3 Notation

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

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

1.4 Earlier works

In this section, we recall the generalized algorithm for computing short discrete logarithms in [3] as the purpose of this paper is to improve upon it.

Given a generator g of a finite cyclic group of order r and a group element $x = [\, d \,]\, g$ where d is such that $0 < d < 2^m \lll r$, the generalized algorithm computes the logarithm $d = \log_g x$ by inducing and observing the system

$$|\, \Psi \,\rangle = \frac{1}{2^{2\ell+m}} \sum_{a,\, j\,=\,0}^{2^{\ell+m}-1} \; \sum_{b,\, k\,=\,0}^{2^{\ell}-1} \exp\left[\frac{2\pi i}{2^{\ell+m}} \, (aj + 2^m bk) \right] |\, j, k, [\, e = a \, - \, bd \,]\, g \,\rangle$$

yielding a pair (j, k) where j and k are integers on the intervals $0 \leq j < 2^{\ell+m}$ and $0 \leq k < 2^{\ell}$, respectively, and some group element $y = [\, e \,]\, g \in \mathbb{G}$.

Above $\ell \approx m/s$ is an integer for $s \geq 1$ a small integer constant that controls the tradeoff between the number of times that the algorithm needs to be executed and the requirements it imposes on the quantum computer.

When observed, the system collapses to $y = [\, e \,]\, g$ and (j, k) with probability

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

where the sums are over all a and b such that $e \equiv a - bd \pmod{r}$. Assuming that $r \geq 2^{\ell+m} + 2^{\ell} d$, this simplifies to $e = a - bd$. Summing over all e, we obtain the probability P of observing the pair (j, k) over all e as

$$\begin{aligned} P &= \frac{1}{2^{2(2\ell+m)}} \sum_e \left| \sum_{b\,=\,b_0(e)}^{b_1(e)-1} \exp\left[\frac{2\pi i}{2^{\ell+m}} \, ((e + bd)j + 2^m bk) \right] \right|^2 \\ &= \frac{1}{2^{2(2\ell+m)}} \sum_e \left| \sum_{b\,=\,b_0(e)}^{b_1(e)-1} \exp\left[\frac{2\pi i}{2^{\ell+m}} \, b \, \{dj + 2^m k\}_{2^{\ell+m}} \right] \right|^2 \end{aligned}$$

where we have introduced the functions $b_0(e)$ and $b_1(e)$ that determine the summation interval for b. For more information on these functions, see Section 2.2.

The above analysis implies that the probability P of observing a given pair (j, k) is determined by its argument α or, equivalently, by its angle θ, where

$$\alpha(j, k) = \{dj + 2^m k\}_{2^{\ell+m}} \quad \text{and} \quad \theta(\alpha) = \frac{2\pi\alpha}{2^{\ell+m}}$$

and the probability

$$P(\theta) = \frac{1}{2^{2(2\ell+m)}} \sum_{e} \left| \sum_{b\,=\,b_0(e)}^{b_1(e)-1} \mathrm{e}^{i\theta b} \right|^2 .$$

In [3], a pair (j, k) is said to be good if $|\,\alpha(j, k)\,| \leq 2^{m-2}$, and it is demonstrated that the quantum algorithm will yield a good pair with probability $p \geq 1/8$. More specifically, lower bounds on both the number of good pairs, and on the probability of observing any specific good pair, are demonstrated.

Given a set of s distinct good pairs, there is a classical post-processing algorithm that recovers d with high probability using lattice-based techniques.

More specifically, the set $\{(j_1, k_1), \ldots, (j_s, k_s)\}$ of s pairs is first used to form a vector $\mathbf{v} = (\{-2^m k_1\}_{2^{\ell+m}}, \ \ldots, \ \{-2^m k_s\}_{2^{\ell+m}}, \ 0) \in \mathbb{Z}^{s+1}$ and an $s+1$-dimensional integer lattice L with basis matrix

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

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

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

is then such that the distance

$$\begin{aligned} R = |\,\mathbf{u} - \mathbf{v}\,| &= \sqrt{\sum_{i=1}^{s} \left(\{dj_i\}_{2^{\ell+m}} + m_i 2^{\ell+m} - \{-2^m k_i\}_{2^{\ell+m}} \right)^2 + d^2} \\ &= \sqrt{\sum_{i=1}^{s} \underbrace{\{dj_i + 2^m k_i\}_{2^{\ell+m}}^2}_{\alpha_i^2} + d^2} \leq 2^m \sqrt{\frac{s}{2^4} + 1} \end{aligned}$$

as $|\,\alpha_i\,| \leq 2^{m-2}$ for good pairs (j, k). By enumerating all vectors in L within distance R of $\mathbf{v}$ one may thus recover $\mathbf{u}$ and d as the last component of $\mathbf{u}$.

For small s, the enumeration is expected to be feasible to perform in practice, as the dimension of L is small allowing a reduced basis to be computed, and as only a few vectors need to be enumerated. To understand why the latter claim holds, note that the volume of a hypersphere of radius R in $D = s + 1$ dimensions is

$$V_D(R) = \frac{\pi^{D/2}}{\Gamma(\frac{D}{2} + 1)} R^D \leq \frac{\pi^{(s+1)/2}}{\Gamma(\frac{s+1}{2} + 1)} \, 2^{(s+1)m} \left(\frac{s}{2^4} + 1 \right)^{(s+1)/2} \tag{1}$$

where Γ is the Gamma function. The volume of the fundamental parallelepiped in L is $\det L = 2^{(\ell+m)s} = 2^{(s+1)m}$ and this volume contains a single lattice vector.

Heuristically, we therefore expect to enumerate up to approximately

$$\frac{V_D(R)}{\det L} \leq \frac{\pi^{(s+1)/2}}{\Gamma(\frac{s+1}{2} + 1)} \left(\frac{s}{2^4} + 1 \right)^{(s+1)/2}$$

vectors in L to recover $\mathbf{u}$ from $\mathbf{v}$. The exact number depends on the placement of the hypersphere in $\mathbb{Z}^D$ and on the shape of the fundamental parallelepiped in L.

As we do not know how to efficiently recognize good pairs, the quantum algorithm has to be executed cs times for some constant $c \approx 1/p = 8$ to generate a set of cs pairs. All subsets of s pairs from these cs pairs are solved for d using the above classical post-processing algorithm. With high probability, at least one of these subsets are expected to consist of s distinct good pairs and to yield d.

The main drawbacks of the above approach are that the complexity of solving for d is exponential in the tradeoff factor s, restricting the achievable tradeoff, and that the quantum algorithm has to be executed cs times.

1.5 Our contributions

One of our key contributions in this paper is to demonstrate that if the quantum algorithm is run $n \geq s$ times, and if all n pairs thus produced are included in an $n + 1$-dimensional lattice L on the above form, then the number of vectors that need to be enumerated in this lattice to recover $\mathbf{u}$ from $\mathbf{v}$ decreases in n.

If n is selected sufficiently large there will hence only be one vector in L within distance R of $\mathbf{u}$. Mapping $\mathbf{v}$ to the closest vector in L using Babai's [1] nearest plane algorithm is then sufficient to immediately recover $\mathbf{u}$ and d without even having to enumerate the lattice. The exhaustive enumeration of exponential in s many $s + 1$-dimensional lattices may thus be avoided, at the expense of reducing a single $n + 1$-dimensional lattice and applying Babai's algorithm.

For cryptographically relevant combinations of m and s, we provide concrete estimates of the number of runs n required to allow $\mathbf{u}$ and d to be recovered from $\mathbf{v}$ given a bound on the number of vectors that we at most accept to enumerate. We verify our estimates by simulating the quantum algorithm and solving sets of n simulated outputs for d using the post-processing algorithm.

To compute the estimates, and to simulate the quantum algorithm for known logarithms, we first analytically derive a closed-form expression for the probability $\Phi(\alpha)$ of the quantum algorithm yielding a pair with argument α. By numerically summing $\Phi(\alpha)$ over partial intervals in α, we then construct a high resolution histogram for $\Phi(\alpha)$ and use it to sample arguments α and pairs (j, k) representative of those that would be yielded by the quantum algorithm. This simulator, and the analysis of the probability distribution, are in themselves key contributions.

The analysis shows the probability of observing a good pair, by the definition in [3], to in general be above 3/10. It is hence much greater than the lower bound of 1/8 given in [3] would imply. More importantly, even if the pair returned is not good, it will still be close to being good. The new post-processing algorithm leverages this fact, and our improved understanding of the probability distribution.

1.6 Overview

The remainder of this paper is structured as follows:

We derive a closed-form expression for $\Phi(\alpha)$ in Section 2. By summing $\Phi(\alpha)$ over intervals in α, we construct a high-resolution histogram for the distribution induced by $\Phi(\alpha)$ in Section 3, and describe how the histogram may be sampled to simulate the quantum algorithm. In Section 4 we proceed to describe our new post-processing algorithm in full detail, to estimate n as a function of m and s, and to verify these estimates by performing simulations. We conclude the paper in Section 5. In appendix A, we quantify the advantage achieved by our algorithm when targeting RSA and Diffie-Hellman with short exponents.

2 Deriving closed-form expressions

In this section, we derive closed-form expressions for the probability $P(\theta(\alpha))$ of observing a specific pair (j, k) with angle θ, or equivalently, with argument α.

Furthermore, we count the number of pairs $N(\alpha)$ with argument α and use it to derive closed-form expressions for the probability $\Phi(\alpha)$ of observing any one of the $N(\alpha)$ pairs (j, k) with argument α.

2.1 The probability of observing (j, k) and e

As explained in Section 1.4, the probability of observing a pair with angle θ is

$$P(\theta) = \frac{1}{2^{2(2\ell+m)}} \sum_{e} \left| \sum_{b\,=\,b_0(e)}^{b_1(e)-1} \mathrm{e}^{i\theta b} \right|^2 = \frac{1}{2^{2(2\ell+m)}} \sum_{e} \underbrace{\left| \sum_{b\,=\,0}^{\#b(e)-1} \mathrm{e}^{i\theta b} \right|^2}_{\zeta(\theta,\,\#b(e))}$$

where $\#b(e) = b_1(e) - b_0(e)$. If $\theta = 0$, then

$$\zeta(0, \#b(e)) = \left| \sum_{b\,=\,0}^{\#b(e)-1} \mathrm{e}^{i\theta b} \right|^2 = \left| \sum_{b\,=\,0}^{\#b(e)-1} 1 \right|^2 = (\#b(e))^2.$$

Otherwise, if $\theta \neq 0$ then

$$\zeta(\theta, \#b(e)) = \left| \sum_{b\,=\,0}^{\#b(e)-1} \mathrm{e}^{i\theta b} \right|^2 = \left| \frac{\mathrm{e}^{i\theta\,\#b(e)} - 1}{\mathrm{e}^{i\theta} - 1} \right|^2 = \frac{1 - \cos(\theta\,\#b(e))}{1 - \cos\theta}.$$

The closed-form expressions for $\zeta(\theta, \#b(e))$ enable us to exactly compute the probability of observing a specific pair with angle θ for a specific e in terms of the angle and the length $\#b(e)$ of the summation interval in b for this e.

In the next section, we analyze the function $\#b(e)$. We proceed in Section 2.3 to sum $\zeta(\theta, \#b(e))$ over all e to derive a closed-form expression for $P(\theta)$.

2.2 The summation interval for a given e

In Section 1.4, we defined $e = a - bd$, where $0 < d < 2^m$ and $0 \leq a < 2^{\ell+m}$ and $0 \leq b < 2^\ell$. This implies that e is an integer on the interval

$$-(2^\ell - 1)d \leq e = a - bd < 2^{\ell+m}. \qquad (2)$$

Divide the interval for e into three regions, and denote these regions A, B and C, respectively. Define the middle region B to be the region in e where $\#b(e) = 2^\ell$ or equivalently $0 = b_0(e) \leq b < b_1(e) = 2^\ell$. Then by (2) region B spans $0 \leq e < 2^{\ell+m} - (2^\ell - 1)d$. This is easy to see, as for all b on the interval $0 \leq b < 2^\ell$ there must exist an a on the interval $0 \leq a < 2^{\ell+m}$ such that $e = a - bd$.

It follows that region A to the left of B spans $-(2^\ell - 1)d \leq e < 0$ and that region C to the right of B spans $2^{\ell+m} - (2^\ell - 1)d \leq e < 2^{\ell+m}$. Regions A and C are hence both of length $(2^\ell - 1)d$. Regions A and C are furthermore both divided into $2^\ell - 1$ plateaus of length d, as there is, for each multiple of d that is subtracted from e in these regions, one fewer value of b for which there exists an a such that $e = a - bd$. The situation that arises is depicted in Fig. 1.

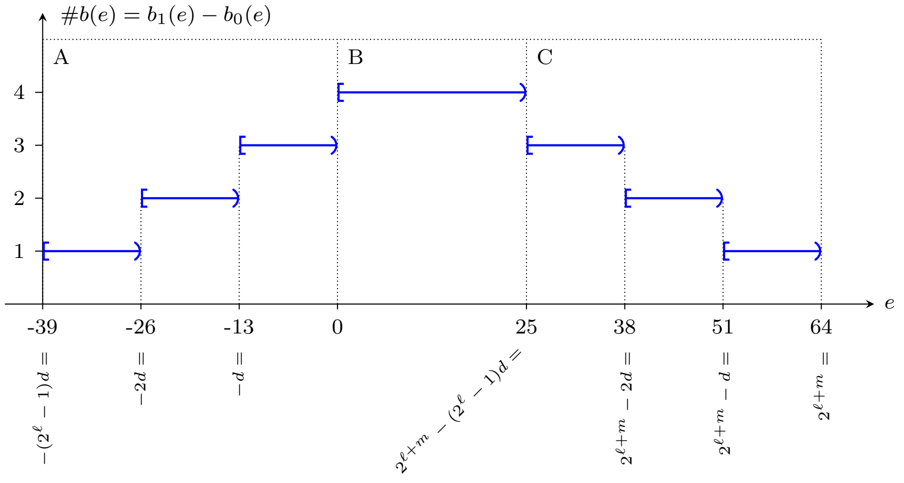

Figure 1: The interval length $\#b(e) = b_1(e) - b_0(e)$ for $m = 4$, $\ell = 2$ and $d = 13$. A similar situation arises irrespective of how these parameters are selected.

2.3 The probability of observing (j, k) over all e

We are now ready to sum $\zeta(\theta, \#b(e))$ over all e to express the probability

$$P(\theta) = \frac{1}{2^{2(m+2\ell)}} \sum_{e\,=\,-(2^\ell-1)d}^{2^{\ell+m}-1} \zeta(\theta, \#b(e))$$

on closed form by first noting that if the sum is split into partial sums over the regions A, B and C, respectively, it follows from the previous two sections that the sums over regions A and C must yield the same contribution.

Furthermore, region B is rectangular, and each plateau in regions A and C is rectangular. Closed-form expressions for the contribution from these rectangular regions may be trivially derived.

This implies that

$$2^{2(m+2\ell)}\, P(\theta) = \underbrace{\sum_{e\,=\,-(2^\ell-1)d}^{-1} \zeta(\theta, \#b(e))}_{\text{region A}} + \underbrace{\sum_{e\,=\,0}^{2^{\ell+m}-(2^\ell-1)d-1} \zeta(\theta, \#b(e))}_{\text{region B}} + \underbrace{\sum_{e\,=\,2^{\ell+m}-(2^\ell-1)d}^{2^{\ell+m}-1} \zeta(\theta, \#b(e))}_{\text{region C}} =$$

$$\underbrace{(2^{\ell+m} - (2^\ell - 1)d) \cdot \zeta(\theta, 2^\ell)}_{\text{region B}} + \underbrace{2d \cdot \sum_{\#b=1}^{2^\ell-1} \zeta(\theta, \#b)}_{\text{regions A and C}}.$$

If $\theta = 0$ then

$$\sum_{\#b=1}^{2^\ell-1} \zeta(0, \#b) = \sum_{\#b=1}^{2^\ell-1} (\#b)^2 = \frac{2^\ell}{6}(2^\ell - 1)(2^{\ell+1} - 1)$$

so the probability of observing the pair (j, k) over all e is

$$P(0) = \frac{1}{2^{2(m+2\ell)}} \cdot \left((2^{\ell+m} - (2^\ell - 1)d) \cdot 2^{2\ell} + \frac{2^\ell d}{3}(2^\ell - 1)(2^{\ell+1} - 1) \right).$$

Otherwise, if $\theta \neq 0$ then

$$\begin{aligned}\sum_{\#b=1}^{2^\ell-1} \zeta(\theta, \#b) &= \sum_{\#b=1}^{2^\ell-1} \frac{1 - \cos(\theta \cdot \#b)}{1 - \cos\theta} \\ &= \frac{1}{1 - \cos\theta}\left[(2^\ell - 1) - \frac{1}{2}\left(\frac{\cos((2^\ell - 1)\theta) - \cos 2^\ell\theta}{1 - \cos\theta} - 1 \right) \right]\end{aligned}$$

so the probability of observing the pair (j, k) over all e is

$$\begin{aligned}P(\theta) = \frac{1}{2^{2(m+2\ell)}} \cdot \frac{1}{1 - \cos\theta} \cdot \Bigg(&(2^{\ell+m} - (2^\ell - 1)d) \cdot (1 - \cos 2^\ell\theta) + \\ &2d \cdot \left[(2^\ell - 1) - \frac{1}{2}\left(\frac{\cos((2^\ell - 1)\theta) - \cos 2^\ell\theta}{1 - \cos\theta} - 1 \right) \right] \Bigg).\end{aligned}$$

2.4 Identifying and counting pairs (j, k) with argument α

In this section we identify and count all pairs (j, k) that yield α.

Definition 2.1. *Let κ denote the greatest integer such that 2^κ divides d.*

Definition 2.2. *An argument α is said to be admissible if there exists a pair $(j, k) \in \mathbb{Z}^2$ where $0 \leq j < 2^{\ell+m}$ and $0 \leq k < 2^\ell$ such that $\alpha = \{dj + 2^m k\}_{2^{\ell+m}}$.*

Claim 2.1. *All admissible $\alpha = \{dj + 2^m k\}_{2^{\ell+m}}$ are multiples of 2^κ.*

Proof. As $2^\kappa \mid d < 2^m$ and the modulus is a power of two the claim follows. ■

Lemma 2.1. *The set of integer pairs (j, k) on $0 \leq j < 2^{\ell+m}$ and $0 \leq k < 2^\ell$ that yield the admissible argument α is given by*

$$j = \left(\frac{\alpha - 2^m k}{2^\kappa} \left(\frac{d}{2^\kappa} \right)^{-1} + 2^{\ell+m-\kappa}\, t \right) \bmod 2^{\ell+m}$$

as t runs trough all integers on $0 \leq t < 2^\kappa$ and k runs trough all integers on $0 \leq k < 2^\ell$. Each admissible argument α hence occurs with multiplicity $2^{\ell+\kappa}$.

Proof. As $\alpha \equiv dj + 2^m k \bmod 2^{\ell+m}$, it follows by solving for j that

$$j = ((\alpha - 2^m k)d^{-1} + 2^{\ell+m-\kappa}\, t) \bmod 2^{\ell+m}$$

for k on $0 \leq k < 2^\ell$ and t on $0 \leq t < 2^\kappa$, and so the lemma follows. ■

2.5 The probability of observing a pair (j, k) with argument α

It follows from the above analysis that the probability $\Phi(\theta(\alpha))$ of observing one of the pairs (j, k) with argument α is $\Phi(\alpha) = N(\alpha) \cdot P(\theta(\alpha))$ where

$$N(\alpha) = \begin{cases} 2^{\ell+\kappa} & \text{if } \alpha \text{ is admissible} \\ 0 & \text{otherwise.} \end{cases}$$

2.6 The notion of t-good pairs

In this section, we introduce the notion of t-good pairs, prove upper bounds on the probability of obtaining a t-good pair, and show how the pairs are distributed in t.

Definition 2.3. *A pair (j, k) is said to be t-good, for t an integer, if*

$$\begin{cases} 2^{t-1} \leq |\,\alpha(j, k)\,| < 2^t & \textit{when} \quad 0 < t < \ell + m \\ \alpha(j, k) = 0 & \textit{when} \quad t = 0. \end{cases}$$

Definition 2.4. *Let $\rho(t)$ denote the probability of observing a t-good pair.*

Lemma 2.2. *For $0 < t < \ell + m$, the probability $\rho(t) < 2^{2-|m-t|}$.*

Proof. For $0 < t < \ell + m$, the probability

$$\rho(t) = \sum_{(j,\,k)} \Phi(\alpha(j, k)) = \frac{1}{2^{2(m+2\ell)}} \sum_{\alpha} N(\alpha) \sum_{e} \zeta(\theta(\alpha), \#b(e))$$

where the first sum is over all pairs (j, k) such that $0 \leq j < 2^{\ell+m}$, $0 \leq k < 2^\ell$ and $2^{t-1} \leq |\,\alpha(j,\,k)\,| < 2^t$. The second sum is over at most $2^{t-\kappa}$ admissible arguments α that each occurs with multiplicity $N(\alpha) = 2^{\ell+\kappa}$. The third sum is over at most $2^{\ell+m+1}$ values of e. As $\zeta(\theta, \#b(e)) \leq 2^{2\ell}$, this implies

$$\rho(t) \leq \frac{2^{t-\kappa} \cdot 2^{\ell+\kappa} \cdot 2^{\ell+m+1} \cdot 2^{2\ell}}{2^{2(m+2\ell)}} = 2^{1-(m-t)}.$$

Furthermore, as $1 - \cos(\theta \cdot \#b(e)) \leq 2$, and as $1 - \cos\theta \geq \theta^2/5$ for $|\,\theta\,| \leq \pi$,

$$\zeta(\theta, \#b(e)) = \frac{1 - \cos(\theta \cdot \#b(e))}{1 - \cos\theta} \leq \frac{10}{\theta^2}.$$

As $|\,\theta(\alpha)\,| = 2\pi\,|\,\alpha\,|\,/2^{\ell+m} \geq 2^t\pi/2^{\ell+m}$, this implies

$$\rho(t) \leq \frac{2^{t-\kappa} \cdot 2^{\ell+\kappa} \cdot 2^{\ell+m+1}}{2^{2(m+2\ell)}} \cdot \frac{10}{\theta^2} \leq \frac{20}{\pi^2} \cdot 2^{m-t} < 2^{2+(m-t)}$$

and so the lemma follows. ■

Lemma 2.2 shows that the probability mass is concentrated on the t-good pairs for $t \approx m$, except for very large κ as $t = 0$ then attracts a non-negligible fraction of the probability mass. The situation that arises is visualized in Fig. 2.

In the region denoted A in Fig. 2, all logarithms d are odd so κ is zero. As d decreases from $2^m - 1$ to $2^{m-1} + 1$, the probability mass is redistributed towards slightly smaller values of t. This reflects the fact that it becomes easier to solve for

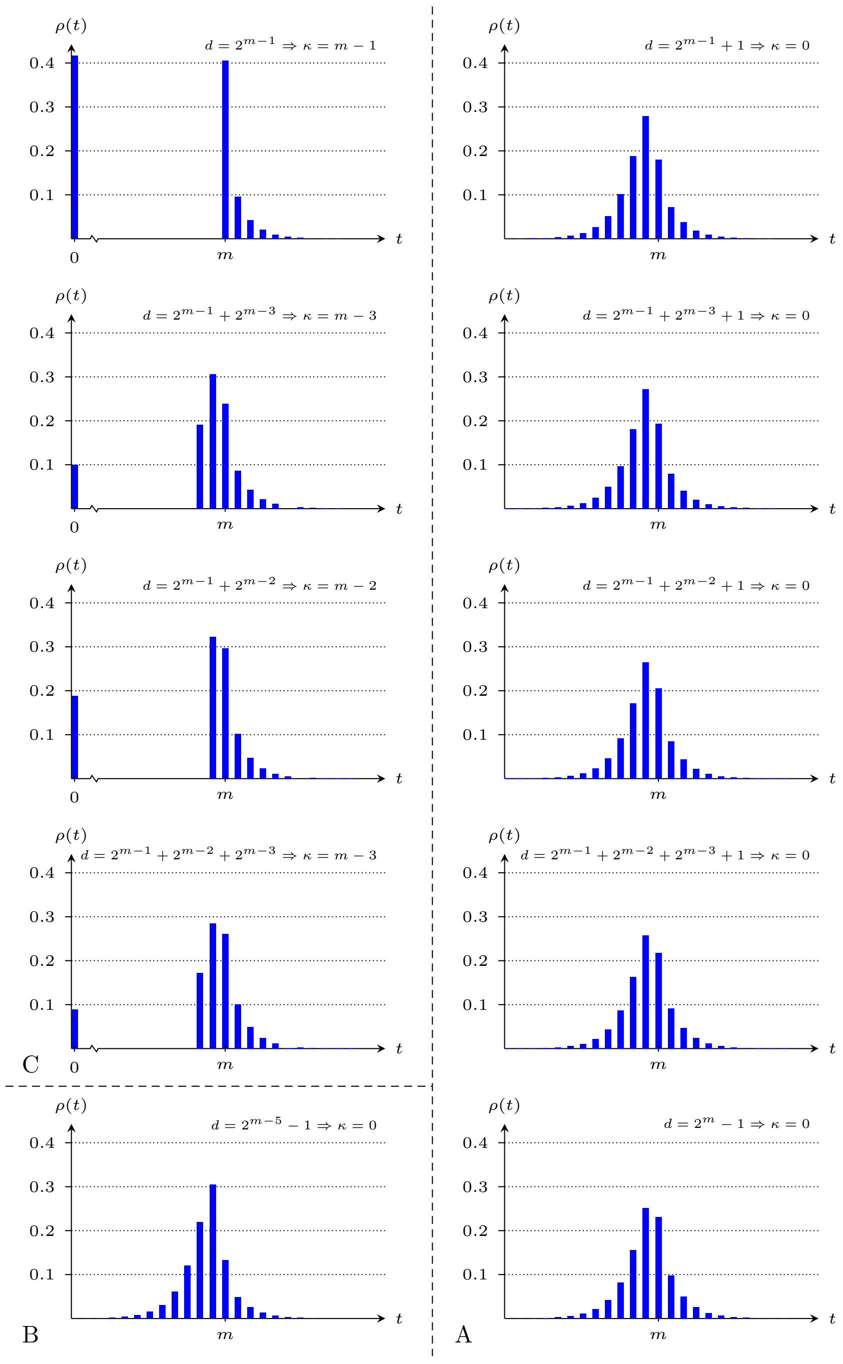

Figure 2: Histograms for $\rho(t)$ for various d and $m = \ell = 256$. Essentially the same histograms arise for any permissible combination of m and ℓ.

d as d decreases in size in relation to 2^m. The redistributive effect weakens if d is further decreased: The histogram in region B where $d = 2^{m-5} - 1$ is representative of the distribution that arises when d is smaller than 2^m by a few orders of magnitude. Further decreasing d has no significant effect on the distribution.

In the region denoted C in Fig. 2, all logarithms d are divisible by a great power of two to illustrate the redistribution that occurs when κ is close to maximal. As the admissible arguments are multiples of 2^κ, it follows that $\rho(t) = 0$ for $0 < t < \kappa$, so the corresponding histogram entries are forced to zero and the probability mass redistributed. This effect is only significant for very large κ; it does not occur in practice for cryptographically relevant problem instances.

Based upon the above analysis and Fig. 2, we conjecture the probability p of observing a good pair to, in general, be considerably greater than the lower bound of $p \geq 1/8$ in [2, 3] guarantees. In general, it should be safe to assume a lower bound of $p \geq 3/10$ for random d on $2^{m-1} \leq d < 2^m$.

3 Simulating the quantum algorithm

In this section, we combine results from the previous sections to construct a high-resolution histogram for the probability distribution for given d. Furthermore, we describe how the histogram may be sampled to simulate the quantum algorithm.

3.1 Constructing the histogram

To construct the high-resolution histogram, we divide the argument axis into regions and subregions and integrate $\Phi(\alpha)$ numerically in each subregion.

First, we subdivide the negative and positive sides of the argument axis into $30 + \mu$ regions where $\mu = \min(\ell - 2, 11)$. Each region thus formed may be uniquely identified by an integer η by requiring that for all α in the region

$$2^{|\eta|} \leq |\,\alpha\,| \leq 2^{|\eta|+1} \qquad \text{and} \qquad \mathrm{sgn}(\alpha) = \mathrm{sgn}(\eta)$$

where $m - 30 \leq |\,\eta\,| < m + \mu - 1$. Then, we subdivide each region into subregions identified by an integer ξ by requiring that for all α in the subregion

$$2^{|\eta| + \xi/2^\nu} \leq |\,\alpha\,| \leq 2^{|\eta| + (\xi+1)/2^\nu}$$

for ξ an integer on $0 \leq \xi < 2^\nu$ and ν a resolution parameter. In this paper, we fix $\nu = 11$ to obtain a high degree of accuracy in the tail of the histogram.

For each subregion, we compute the approximate probability mass contained within the subregion by applying Simpson's rule, followed by Richardson extrapolation to cancel the linear error term. Simpson's rule is hence applied $2^\nu(1 + 2^\nu)$ times in each region. Each application requires $\Phi(\alpha)$ to be evaluated in up to three points (the two endpoints and the midpoint). As the distribution is symmetric around zero, we need only compute one side in practice. When evaluating $\Phi(\alpha)$, we divide the result by 2^κ to account for the density of admissible pairs.

This approach to constructing the histogram is only valid if κ is small in relation to m, as we must otherwise account for which α are admissible. For small κ, this is not necessary, as the variation in $\Phi(\theta(\alpha))$ is negligible for small variations in α.

3.2 Sampling the distribution

To sample an argument α from the histogram for the distribution, we first sample a subregion and then sample α uniformly at random from the set of all admissible arguments in this subregion. To sample the subregion, we first order all subregions in the histogram by probability, and compute the cumulative probability up to and including each subregion in the resulting ordered sequence. Then, we sample a pivot uniformly at random from $[0, 1)$, and return the first subregion in the ordered sequence for which the cumulative probability is greater than or equal to the pivot. Sampling fails if the pivot is greater than the total cumulative probability.

To sample an integer (j, k) from the distribution, we first sample an admissible argument α as described above, and then sample (j, k) uniformly at random from the set of all integer pairs (j, k) yielding α using Lemma 2.1. More specifically, we first sample an integer t uniformly at random from the interval $0 \leq t < 2^\kappa$ and then compute (j, k) from α and t as described in Lemma 2.1.

4 Our improved post-processing algorithm

In this section, we introduce our new post-processing algorithm. It allows d to be recovered from a set $\{(j_1, k_1), \ldots, (j_n, k_n)\}$ of $n > s$ pairs produced simply by executing the quantum algorithm n times.

In analogy with the original post-processing algorithm, the set of n pairs is used to form a vector $\mathbf{v} = (\{-2^m k_1\}_{2^{\ell+m}}, \ldots, \{-2^m k_n\}_{2^{\ell+m}}, 0) \in \mathbb{Z}^D$ and a D-dimensional integer lattice L with basis matrix

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

where $D = n + 1$. For some constants $m_1, \ldots, m_n \in \mathbb{Z}$, the vector

$$\mathbf{u} = (\{dj_1\}_{2^{\ell+m}} + m_1 2^{\ell+m}, \ldots, \{dj_n\}_{2^{\ell+m}} + m_n 2^{\ell+m}, d) \in L$$

is such that the distance

$$R = |\,\mathbf{u} - \mathbf{v}\,| = \sqrt{\sum_{i=1}^{n} \left(\{dj_i\}_{2^{\ell+m}} + m_i 2^{\ell+m} - \{-2^m k_i\}_{2^{\ell+m}}\right)^2 + d^2}$$

$$= \sqrt{\sum_{i=1}^{n} \underbrace{\{dj_i + 2^m k_i\}^2_{2^{\ell+m}}}_{\alpha_i^2} + d^2} = \sqrt{\sum_{i=1}^{n} \alpha_i^2 + d^2}\,.$$

This implies that $\mathbf{u}$ and hence d may be found by enumerating all vectors in L within a D-dimensional hypersphere of radius R centered on $\mathbf{v}$. The volume of such a hypersphere is $V_D(R)$ as defined in equation 1.

For comparison, the fundamental parallelepiped in L contains a single lattice vector and is of volume $\det L = 2^{(\ell+m)n}$. Heuristically, we therefore expect the hypersphere to contain approximately $v = V_D(R) \,/\, \det L$ lattice vectors. The exact number depends on the placement of the hypersphere in $\mathbb{Z}^D$ and on the shape of the fundamental parallelepiped in L.

Assuming v to be sufficiently small for it to be computationally feasible to enumerate all lattice vectors in the hypersphere in practice, the above algorithm may be used to recover d. As the volume quotient v decreases in n, the number of vectors that need to be enumerated may be reduced by running the quantum algorithm more times and including the resulting pairs in L. However, there are limits to the number of pairs that may be included in L, as a reduced basis must be computed to enumerate L^j, and the complexity of computing such a basis grows rapidly in the dimension of L^j.

In this section, we show how to heuristically estimate the minimum number of runs n required to solve a specific known problem instance for d with minimum success probability q, for a given tradeoff factor s, and for a given bound on the number of vectors v that we at most accept to enumerate in L.

4.1 Estimating the minimum n required to solve for d

The radius R depends on the pairs (j_i, k_i) via the arguments α_i. For fixed n and fixed probability q, we may estimate the minimum radius $\widetilde{R}$ such that

$$\Pr\left[R = \sqrt{\sum_{i=1}^{n} \alpha_i^2 + d^2} \leq \widetilde{R} \right] \geq q$$

by sampling α_i from the probability distribution as described in Section 4.2. Then

$$\Pr\left[v = \frac{V_D(R)}{\det L} \leq \frac{V_D(\widetilde{R})}{2^{(\ell+m)n}} \right] \geq q, \tag{3}$$

providing an heuristic bound on the number of lattice vectors v that at most have to be enumerated, that holds with probability at least q.

Given an upper limit on the number of lattice vectors that we accept to enumerate, equation (3) may be used as an heuristic to estimate the minimum value of n such that v is below this limit with probability at least q.

To compute the estimate in practice, we use the heuristic to compute an upper bound on v for $n = 1, 2, \ldots$ and return the minimum n for which the bound is below the limit on the number of vectors that we accept to enumerate.

As the volume quotient v decreases by approximately a constant factor for every increment in n, the minimum n may be found efficiently via interpolation once the heuristic bound on v has been computed for a few values of n.

4.2 Estimating $\widetilde{R}$

To estimate $\widetilde{R}$ for m, s and n, explicitly known d, and a given target success probability q, we sample N sets of n arguments $\{\alpha_1, \ldots, \alpha_n\}$ from the probability distribution as described in Section 3.2. For each set, we compute R, sort the resulting list of values in ascending order, and select the value at index $\lceil (N-1)\,q \rfloor$ to arrive at our estimate for $\widetilde{R}$. The constant N controls the accuracy of the estimate. Assuming N to be sufficiently large in relation to q, and to the variance of the arguments, this approach yields sufficiently good estimates.

4.2.1 On sampling failures

The sampling of argument pairs may fail. This occurs when the pair being sampled is not in the regions of the argument axis covered by the histogram constructed for the probability distribution. If the sampling of at least one argument in a set fails, we err on the side of caution and over-estimate $\widetilde{R}$ by letting $R = \infty$ for the set. The entries for the failed sets will then all be sorted to the end of the lists. If the value of $\widetilde{R}$ selected from the sorted list is ∞ no estimate is produced.

4.3 Results and analysis

To estimate n as a function of m and s, and to verify the estimates in simulations, we fix $q = 0.99$ and consider the hard case $d = 2^m - 1$.

For relevant combinations of m and s, we let $\ell = \lceil m/s \rceil$, fix $N = 10^6$ when estimating $\widetilde{R}$, and record the smallest $n > s$ for which the volume quotient $v < 2$. For some m and s, we verify the estimate by sampling $M = 10^3$ sets of n pairs $\{(j_1, k_1), \ldots, (j_n, k_n)\}$ and solving each set for d with the post-processing algorithm. If d is thus recovered, the verification succeeds, otherwise it fails. We record the smallest $n > s$ such that at most $M(1 - q) = 10$ verifications fail.

	logarithm size m						
tradeoff factor s	128	256	512	1024	2048	4096	8192
1	2	2	2	2	2	2	2
2	3	3	3	3	3	3	3
3	4	4	4	4	4	4	4
4	6	5	5	5	5	5	5
5	7 / 8	6	6	6	6	6	6
6	10	8	7	7	7	7	7
7	13	10 / 9	8	8	8	8	8
8	18	11	10	9	9	9	9
10	– / 32	15	12	11	11	11	11
20	–	– / 71	32 / 31	24	22	21	21
30	–	–	– / 60	41 / 40	35	32	31
40	–	–	–	– / 62	50 / 48	45 / 44	42
50	–	–	–	–	– / 65	58 / 57	54 / 53
60	–	–	–	–	–	– / 69	65
70	–	–	–	–	–	–	– / 76

Table 1: The estimated (A) and simulated (B) number of pairs n required for $\mathbf{u}$ to be the closest vector to $\mathbf{v}$ in L allowing d to be recovered via Babai's algorithm. If A = B, we only print A, otherwise we print B/A. Dash indicates no information.

Table 1 was produced by executing these procedures. To reduce the lattice bases, the block Korkin-Zolotarev (BKZ) algorithm, introduced by Schnorr [4, 5], was employed, as implemented in fpLLL 5.2, with default parameters and a block size of $\min(10, n+1)$ for all combinations of m, s and n. For these parameter choices, a basis takes at most minutes to reduce and solve for d in a single thread.

The estimated values of n are verified by the simulations, except when v is close to two, in which case the estimated and simulated n may differ slightly. Note that the variation becomes stronger when the factor whereby v increases or decreases for each increment of decrement of n is small in relation to two.

Compared to the post-processing algorithm originally proposed, the new post-processing algorithm efficiently achieves considerably better tradeoffs with practical time complexity for cryptographically relevant parameter choices. It furthermore requires considerably fewer quantum algorithm runs. Asymptotically, the number of runs required n tends to $s + 1$ as m tends to infinity for fixed s, when we require a solution to be found without enumerating the lattice.

4.4 Further improvements

Above, we fixed a high minimum success probability $q = 0.99$ and considered the hard case where $d = 2^m - 1$. Furthermore, we required that mapping $\mathbf{v}$ to the closest vector in L should yield $\mathbf{u}$ without having to enumerate vectors in L.

In practice, some of these choices may be relaxed: Instead of requiring $\mathbf{u}$ to be the closest vector to $\mathbf{v}$ in L, we may enumerate all vectors in a hypersphere of limited radius centered on $\mathbf{v}$. In cryptographic applications, the logarithm d may in general be assumed to be randomly selected on $2^{m-1} \leq d < 2^m$. If not, the logarithm may be randomized; solve $x \odot [\, c \,]\, g$ for $d + c$ with respect to the basis g.

Other options include selecting m slightly greater than the bit length of d, excluding subsets of pairs (j, k) from the lattice to eliminate pairs with large $|\, \alpha(j, k) \,|$, and centering d around zero by admitting negative logarithms.

5 Summary and conclusion

We have introduced a new efficient post-processing algorithm that is practical for greater tradeoff factors, and in general requires considerably fewer quantum algorithm runs, than the original post-processing algorithm. To estimate the number of runs required, we have analyzed the probability distribution induced by the quantum algorithm and developed a method of simulating the quantum algorithm.

With the new post-processing, our quantum algorithm achieves an advantage over Shor's algorithms, not only in each individual run, but also overall, when targeting cryptographically relevant instances of RSA and Diffie-Hellman with short exponents, depending on how parameters are selected. The reader is referred to appendix A for a detailed analysis and quantification of the advantage.

Acknowledgments

I am grateful to Johan Håstad for valuable comments and advice, to Lennart Brynielsson, and other colleagues, for their comments on early versions of this manuscript, and to past reviewers. Funding and support for this work was provided by the Swedish NCSA that is a part of the Swedish Armed Forces.

Computations were performed on Tegner at PDC at KTH. Access was provided by the Swedish National Infrastructure for Computing (SNIC).

References

[1] L. Babai, On Lovász' lattice reduction and the nearest lattice point problem, *Combinatorica* 6(1) (1986), 1–13.

[2] M. Ekerå, *Modifying Shor's algorithm to compute short discrete logarithms*, IACR ePrint Archive, report 2016/1128 (2016).

[3] M. Ekerå and J. Håstad, Quantum algorithms for computing short discrete logarithms and factoring RSA integers, in: *PQCrypto*, Lecture Notes in Comput. Sci. 10346, Springer, Cham (2017), 347–363.

[4] C.-P. Schnorr, A hierarchy of polynomial time lattice basis reduction algorithms, *Theoret. Comput. Sci.* 53(2—3) (1987), 201–224.

[5] C.-P. Schnorr and M. Euchner, Lattice basis reduction: Improved practical algorithms and solving subset sum problems, *Math. Program.* 66(1–3) (1994), 181–199.

[6] J.-P. Seifert, Using fewer qubits in Shor's factorization algorithm via simultaneous Diophantine approximation, in: *CT-RSA*, Lecture Notes in Comput. Sci. 2020, Springer, Berlin Heidelberg (2001), 319–327.

[7] P.W. Shor, Algorithms for Quantum Computation: Discrete Logarithms and Factoring, in: *SFCS*, Proceedings of the 35th Annual Symposium on Foundations of Computer Science, IEEE Computer Society, Washington, DC (1994), 124–134.

[8] P.W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, *SIAM J. Comput.* 26(5) (1997), 1484–1509.

[9] NIST, SP 800-56A: *Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography*, 3rd revision. (2018)

[10] D. Gillmor, RFC 7919: *Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS)*, (2016)

[11] T. Kivinen and M. Kojo, RFC 3526: *More Modular Exponentiation (MODP) Diffie-Hellman groups for Internet Key Exchange.* (2003)

A Our advantage

In this appendix, we analyze our algorithm's advantage when targeting RSA or the Diffie-Hellman key exchange scheme in safe-prime groups with short exponents.

A.1 Diffie-Hellman

The portions of the FIPS 800-56A recommendation [9] that pertain to Diffie-Hellman key exchange in $\mathbb{G} \subset \mathbb{F}_p^*$ were recently revised by NIST.

The previous version mandated the use of randomly selected Schnorr groups and supported moduli p of length up to 2048 bits. The new version supports moduli of length up to 8192 bits. For lengths exceeding 2048 bits, it mandates the use of a fixed set of safe-prime groups, originally developed for TLS and IKE [10, 11].

Migrating from Schnorr groups to safe-prime groups with full length exponents would result in a significant performance penalty. NIST therefore permits the use of short exponents; the exponent length m must be on $2z \leq m \leq \lceil \log_2 r \rceil$, where z is strength level of the group and $r = (p-1)/2$ is the group order.

	strength					in each run		overall	
$\lceil \log_2 p \rceil$	level z	m	s	n	v	#ops	adv	#ops	adv
2048	112	224	1	1	$1.3 \cdot 10^3$	672	6.1	672	6.1
		224	7	10	$6.5 \cdot 10^{-4}$	288	14.2	2880	1.4
3072	128	256	1	1	$1.3 \cdot 10^3$	768	8.0	768	8.0
		256	8	11	$2.4 \cdot 10^{-1}$	320	19.2	3520	1.7
4096	152	304	1	1	$1.3 \cdot 10^3$	912	9.0	912	9.0
		304	9	12	$4.8 \cdot 10^{-1}$	372	22.0	4464	1.8
6144	176	352	1	1	$1.4 \cdot 10^3$	1056	11.6	1056	11.6
		352	10	13	$4.5 \cdot 10^{-2}$	424	29.0	5512	2.2
8192	200	400	1	1	$1.3 \cdot 10^3$	1200	13.7	1200	13.7
		400	11	14	7.5	474	34.6	6636	2.5

Table 2: Sample estimates for the complexity of computing short discrete logarithms in the FIPS 800-56A groups originally developed for TLS and IKE.

In Table 2 we provide estimates of the complexity of computing short discrete logarithms in these widely used safe-prime groups. The estimates were computed as in Section 4.3, for maximal $d = 2^m - 1$ and $\geq 99\%$ success probability, except that we accept to enumerate vectors in L. We tabulate m, s and n, for $s = 1$, and for s the greatest tradeoff factor such that $n - s \leq 3$ with v close to two.

The number of group operations that need to be computed, in each run of the quantum algorithm, and overall in n runs, are tabulated for each tuple m, s and n, along with our advantage, defined as the fraction between $2 \lceil \log_2 r \rceil$, the number of group operations in each run of Shor's algorithm [2,7,8] for general discrete logarithms, and the number of group operations, in each run or overall, in our algorithm. We assume that standard double-and-add exponentiation is used.

Note that the latter comparison is quite generous to Shor as our algorithm has $\geq 99\%$ probability of recovering d after n runs. Multiple runs of Shor's algorithm may be required to achieve a similar bound on the success probability.

As may be seen in Table 2, our algorithm reduces the number of group operations in each run by up to a factor of 34.6 for these m, s and n. It provides an advantage, not only in each run, but also overall, even for large tradeoff factors, due to the new post-processing algorithm being more efficient and requiring fewer runs. We have verified the estimates by post-processing simulated outputs.

A.2 RSA

Let $p, q > 2$ be two large distinct primes of length l bits. Then $N = pq$ is said to be an RSA integer. To factor N into p and q, we propose to do the following:

Pick a random element $g \in \mathbb{Z}_N^*$. Let r denote the unknown order of g. Let $\tilde{p} = (p-1)/2$ and $\tilde{q} = (q-1)/2$. Then r divides $2\tilde{p}\tilde{q} \,/\, \gcd(\tilde{p}, \tilde{q})$. Let $f(N) = (N - 1)/2 - 2^{l-1} = 2\tilde{p}\tilde{q} + \tilde{p} + \tilde{q} - 2^l$. Compute $x = g^{f(N)} \equiv g^d$ for d on $0 \leq d < r$. Then

$d = f(N) \bmod r = \tilde{p} + \tilde{q} - 2^{l-1}$, assuming $r > 2^{l-1}$, so the discrete logarithm d is short. To factor N, it hence suffices to compute d using our algorithm, and to solve $p + q = 2(d + 2^{l-1} + 1)$ and $pq = N$ for p and q.

Recall that in our algorithm for computing short discrete logarithms [3], we require d to be on $0 \leq d < 2^m$, which implies $m = l - 1$. Furthermore, we require $r \geq 2^{\ell+m} + 2^\ell d$, where $\ell = \lceil m/s \rceil$ for s an integer. This implies $s \geq 2$ if the requirement that $r \geq 2^{\ell+m} + 2^\ell d$ is to be respected with high probability.

In Table 3 we provide estimates of the complexity of factoring RSA integers. The estimates were computed as described in Section 4.3, for $\geq 99\%$ success probability and maximal $d = 2^m - 1$. Note that selecting d maximal slightly over-estimates the hardness of factoring random RSA integers. We tabulate m, s and n, for $s = n = 2$, and for s the greatest tradeoff factor such that $n - s \leq 3$ whilst keeping v sufficiently small to avoid having to enumerate the lattice.

The estimates are tabulated as described in Section A.1, except that we compute our advantage in relation to a single run of Shor's order finding algorithm. We have verified the estimates by post-processing simulated quantum algorithm outputs.

					in each run		overall	
$\lceil \log_2 N \rceil$	m	s	n	v	#ops	adv	#ops	adv
2048	1023	2	2	$1.4 \cdot 10^5$	2047	2.0	4094	1.0
	1023	17	20	$3.3 \cdot 10^{-7}$	1145	3.6	22900	0.18
3072	1535	2	2	$1.4 \cdot 10^5$	3071	2.0	6142	1.0
	1535	21	24	$5.7 \cdot 10^{-9}$	1683	3.6	40392	0.15
4096	2047	2	2	$1.4 \cdot 10^5$	4095	2.0	8190	1.0
	2047	24	27	$1.0 \cdot 10^{-10}$	2219	3.7	59913	0.14
6144	3071	2	2	$1.4 \cdot 10^5$	6143	2.0	12286	1.0
	3071	31	34	$8.2 \cdot 10^{-7}$	3271	3.8	111214	0.11
8192	4095	2	2	$1.4 \cdot 10^5$	8191	2.0	16382	1.0
	4095	34	37	$5.1 \cdot 10^{-14}$	4337	3.8	160469	0.10

Table 3: Estimated complexity of factoring RSA integers with $\ell = \lceil m/s \rceil$ and $s \geq 2$ in multiple runs, with $\geq 99\%$ success probability and maximal $d = 2^m - 1$.

As may be seen in Table 3, we obtain an advantage in each run of the algorithm, that translates into a reduction of the size and depth of the quantum circuit.

For $s = 2$, we perform the same number of operations overall in the two runs, as Shor's algorithm does in a single run. Hence, the benefit of reduced circuit size and depth in each individual run comes at no cost in terms of the overall number of operations that need to be performed. For $s > 2$, we obtain a greater advantage in each individual run of the algorithm, at the expense of performing a greater number of operations overall than Shor's algorithm does in a single run.

Compared to the advantage Seifert's algorithm [6] achieves over Shor in each individual run and for a specific tradeoff factor s, our algorithm achieves a further advantage by approximately a factor of two.

A.2.1 Breaking RSA with an advantage in a single run

When using the new post-processing algorithm, we may let $\ell = m - \delta$ for some small δ, instead of letting $\ell = \lceil m/s \rceil$ for s an integer. This tweak allows us to remove the obstacle that prevented us from achieving an advantage over Shor's algorithm in a single run above, without affecting the analysis of the algorithm.

Experiments, in which we select random $N = pq$ and estimate the distribution of orders of elements g selected uniformly at random from $\mathbb{Z}_N^*$ by partially factoring $p-1$ and $q-1$, indicate that selecting $\delta = 20$ results in a very high probability of respecting the requirement that $r \geq 2^{\ell+m} + 2^{\ell} d$.

$\lceil \log_2 N \rceil$	m	ℓ	n	v	#ops	adv
2048	1023	1003	1	$1.4 \cdot 10^9$	3029	1.35
3072	1535	1515	1	$1.4 \cdot 10^9$	4565	1.35
4096	2047	2027	1	$1.4 \cdot 10^9$	6101	1.34
6144	3071	3051	1	$1.4 \cdot 10^9$	9173	1.34
8192	4095	4075	1	$1.4 \cdot 10^9$	12245	1.34

Table 4: Estimated complexity of factoring RSA integers with $\ell = m - \delta$ for $\delta = 20$ in a single run, with $\geq 99\%$ success probability and maximal $d = 2^m - 1$.

In Table 4 we provide estimates for the complexity of factoring RSA integers in a single run of our quantum algorithm with $\ell = m - \delta$ for $\delta = 20$. As evidenced, tweaking ℓ enables us to achieve an advantage over Shor's order finding algorithm in a single run, at the expense of enumerating a moderate number of lattice vectors.

The enumeration is easy to perform in practice. It is facilitated by the lattice being two-dimensional, and by the last component of the vector sought being d. The fact that $0 < d \leq 2^m$ and $d \equiv (N-1)/2 \pmod 2$ may hence be used to prune the enumeration. We have verified that the tweaked algorithm achieves $\geq 99\%$ success probability by post-processing simulated quantum algorithm outputs.