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

Martin Ekerå[*]

November 20, 2017

Abstract

Ekerå and Håstad recently introduced a quantum algorithm for computing short discrete logarithms. To compute an m bit logarithm d this algorithm 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. Increasing s trades work in the quantum algorithm for classical work. However, as good outputs cannot be trivially distinguished, the quantum algorithm needs to be run s/p times and all subsets of s outputs exhaustively post-processed, where p is the probability of observing a good output. In this paper, we introduce an improved post-processing algorithm that removes the need to exhaustively enumerate exponential in s many $s + 1$-dimensional lattices, at the expense of reducing a single $n+1$ dimensional lattice and applying Babai's algorithm. The new algorithm is practical for much greater values of s and in general requires fewer runs than the original algorithm. As a concrete example, for $m = 1024$, $s = 30$ and $n = 40$, the algorithm recovers d within a few seconds with $\approx 99\%$ success probability.

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 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, the author [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.

[*] 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.

This work was subsequently generalized by the author 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 [9] for making tradeoffs in Shor's order finding algorithm; the quantum part of Shor's general factoring algorithm.

The author 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 the author 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 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 effectively trades work in each quantum algorithm run for work in the classical post-processing algorithm. More specifically, the quantum circuit size and depth, the required coherence time and, assuming they are not recycled; the number of control qubits required to implement the algorithm, are all reduced by a constant factor.

To minimize the work performed in each run of the quantum algorithm, s would ideally be selected of size similar to m. However, as good outputs cannot be trivially distinguished, the quantum 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 s and hence the achievable tradeoff.

Our main contribution 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 [1] algorithm. 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 practical for comparatively large s and in general requires 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 short discrete logarithm problem, the logarithm d is known to be smaller than the order of the group by some factor.

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.

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 $-n/2 \leq \{u\}_n < n/2$.
- $\lceil u \rceil$ denotes u rounded upwards to the closest integer.
- $\lceil u \rfloor$ denotes u rounded to the closest integer.
- $\lfloor u \rfloor$ denotes u rounded downwards 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 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

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

$$= \frac{1}{2^{2(2\ell+m)}} \sum_{e} \left| \sum_{b\,=\,b_0(e)}^{b_1(e)-1} \exp\left[\frac{2\pi i}{2^{\ell+m}}\, b \,\{dj + 2^m k\}_{2^{\ell+m}} \right] \right|^2$$

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 Sect. 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

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

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 is said to be good if its argument $\alpha \leq 2^{m-2}$, 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 great 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

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

as the argument $\alpha_i \leq 2^{m-2}$ for good pairs (j, k). By enumerating all vectors in L within distance R of $\mathbf{v}$, see appendix C, it is thus possible to recover $\mathbf{u}$ and d as the last component of $\mathbf{u}$. The enumeration is expected to be feasible to perform in practice for small s, as the dimension $D = s + 1$ of L is small allowing for a reduced basis to be computed, and as the shortest vector in L is expected to be of norm about

$$\sqrt[D]{\left|\det L\right|} = \sqrt[D]{2^{(\ell+m)s}} \approx 2^m$$

which is close to R for small s, implying that there are only a small number of vectors within distance R of $\mathbf{v}$ to enumerate.

As good pairs cannot be trivially distinguished, the quantum algorithm is 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 then solved for d using the above classical post-processing algorithm. With great probability, at least one of these subsets should consist of s distinct good pairs and yield d.

The main drawback with this approach is that the complexity of solving for d is exponential in the tradeoff factor s, restricting the achievable tradeoff.

1.5 Our contributions

Our main contribution in this paper is to demonstrate that if the above 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 distance R between $\mathbf{u}$ and $\mathbf{v}$ decreases in n in relation to the norm of the shortest vector in L.

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 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 relevant combinations of the logarithm length m and tradeoff factor s, we let estimate the value of n required to allow d to be immediately recovered. We verify our estimates by simulating the quantum algorithm and solving sets of n simulated outputs for d using the new post-processing algorithm.

To compute the estimates, and to simulate the quantum algorithm for known logarithms, we 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 construct a high resolution histogram for $\Phi(\alpha)$ and use it to sample arguments and pairs in a manner representative of those yielded by the quantum algorithm.

We simulate the quantum algorithm not only to verify parameter estimates for our new post-processing algorithm, but also to verify claims made in [3] with respect to the original post-processing algorithm. Furthermore, we use the histogram for $\Phi(\alpha)$ to show that the probability p of observing a good pair by the definition in [3] is, in general, considerably greater than $1/8$ as guaranteed by the lower bound in [3]. This implies that the original post-processing algorithm is more efficient than was previously demonstrated.

1.6 Overview

The remainder of this paper is structured as follows: We derive a closed-form expression for $\Phi(\alpha)$ in Sect. 2, we numerically sum $\Phi(\alpha)$ over partial intervals in α in Sect. 3 and we describe how the quantum algorithm may be simulated on a classical computer when the logarithm is known in Sect. 4.

In Sect. 6 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 Sect. 7.

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 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

The probability of observing a pair with angle θ is

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

as was explained in Sect. 1.4. If $\theta = 0$ then the inner sum simplifies to

$$\xi(0, \#b(e)) = \left| \sum_{b\,=\,b_0(e)}^{b_1(e)-1} \mathrm{e}^{i\theta b} \right|^2 = \left| \sum_{b\,=\,b_0(e)}^{b_1(e)-1} 1 \right|^2 = (b_1(e) - b_0(e))^2 = (\#b(e))^2$$

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

$$\xi(\theta, \#b(e)) = \left| \sum_{b\,=\,b_0(e)}^{b_1(e)-1} \mathrm{e}^{i\theta b} \right|^2 = \left| \frac{\mathrm{e}^{i\theta b_1(e)} - \mathrm{e}^{i\theta b_0(e)}}{\mathrm{e}^{i\theta} - 1} \right|^2 = \frac{1 - \cos(\theta \cdot \#b(e))}{1 - \cos\theta}.$$

The closed-form expressions for $\xi(\theta, \#b(e))$ 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 Sect. 2.3 to sum $\xi(\theta, \#b(e))$ over all e to derive a closed-form expression for $P(\theta)$.

2.2 The summation interval for a given e

In Sect. 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

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

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

$$\#b(e) = 2^\ell \quad \Leftrightarrow \quad 0 = b_0(e) \leq b < b_1(e) = 2^\ell.$$

Then by (1) region B spans $0 \leq e < 2^{\ell+m} - (2^\ell - 1)d$. This is easy to since 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 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.

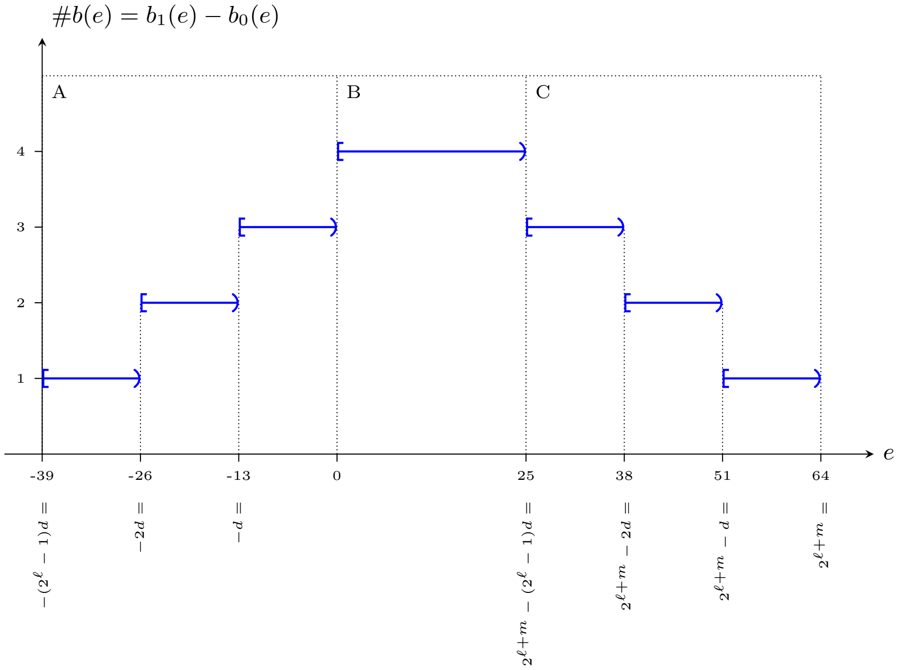

Figure 1: The interval length $\#b(e) = b_1(e) - b_0(e)$ as a function of e. Above $m = 4$, $\ell = 2$ and $d = 13$ but the same situation arises irrespective of these parameter choices.

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

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

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

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

$$\begin{aligned}
\sum_{e\,=\,-(2^\ell-1)d}^{2^{\ell+m}-1} \xi(\theta, \#b(e)) = {} & \underbrace{\sum_{e\,=\,-(2^\ell-1)d}^{-1} \xi(\theta, \#b(e))}_{\text{region A}} \; + \\
& \underbrace{\sum_{e\,=\,0}^{2^{\ell+m}-(2^\ell-1)d-1} \xi(\theta, \#b(e))}_{\text{region B}} \; + \\
& \underbrace{\sum_{e\,=\,2^{\ell+m}-(2^\ell-1)d}^{2^{\ell+m}-1} \xi(\theta, \#b(e))}_{\text{region C}} \; = \\
& \underbrace{(2^{\ell+m} - (2^\ell - 1)d) \cdot \xi(\theta, 2^\ell)}_{\text{region B}} \; + \; \underbrace{2d \cdot \sum_{\#b\,=1}^{2^\ell-1} \xi(\theta, \#b)}_{\text{regions A and C}} .
\end{aligned}$$

If $\theta = 0$ then

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

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

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

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 The number of pairs with argument α

To define the probability of observing some pair with argument α, we now turn our attention to counting the number of pairs (j, k) with argument α.

To perform the count, we demonstrate how the set of all pairs (j, k) with argument α may be computed, as this will anyhow prove necessary in Sect. 4 when simulating the quantum algorithm. More specifically, we first show in Claim 1 that only some arguments are admissible. For admissible arguments α, we then show in Lemma 1 how to compute one or more j for any given k such that the pair (j, k) has argument α up to sign. To conclude, we count all pairs thus generated in Lemma 2 and demonstrate that all pairs have been accounted for.

Claim 1. *For a pair (j, k) where j and k are integers, the argument*

$$\alpha = |\, \{dj + 2^m k\}_{2^{\ell+m}} \,|$$

is either zero or $2^\kappa \mid \alpha$ where κ is the greatest integer such that $2^\kappa \mid d$.

Proof. As $2^\kappa \mid d < 2^m$ it must be that $2^\kappa \mid 2^m$ and hence $2^\kappa \mid \alpha$ when $\alpha \neq 0$. ■

Lemma 1. *Let κ be the greatest integer such that $2^\kappa \mid d$ and let α be an integer on $0 \leq \alpha < 2^{m+\ell}$ such that $2^\kappa \mid \alpha$. For k any integer on $0 \leq k < 2^\ell$ and*

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

the pair (j, k) is distinct and has argument α for r any integer on $0 \leq r < 2^\kappa$.

Proof. For each k there are 2^ℓ distinct values of j up to sign as $2^{\ell+m-\kappa} r \mod 2^{\ell+m}$ is distinct for all integers r on $0 \leq r < 2^\kappa$ so all (j, k) are distinct.

The argument of a pair (j, k) is defined as $\alpha = |\, \{dj + 2^m k\}_{2^{\ell+m}} \,|$. As

$$dj + 2^m k \equiv (\pm\alpha - 2^m k) \underbrace{\frac{d}{2^\kappa} \left(\frac{d}{2^\kappa} \right)^{-1}}_{=1} + 2^{\ell+m} \underbrace{\frac{dr}{2^\kappa}}_{\in \mathbb{Z}} + 2^m k \equiv \pm\alpha \pmod{2^{\ell+m}}$$

the pairs all have argument α and so the lemma follows. ■

Lemma 2. *The number of pairs (j, k) with argument α is*

$$N(\alpha) = \begin{cases} 2^{\kappa+\ell} & \textit{if} \quad \alpha = 0 \\ 2^{\kappa+\ell+1} & \textit{if} \quad 0 < \alpha < 2^{\ell+m-1} \quad \textit{and} \quad 2^\kappa \mid \alpha \\ 2^{\kappa+\ell} & \textit{if} \quad \alpha = 2^{\ell+m-1} \\ 0 & \textit{otherwise} \end{cases}$$

where κ is the greatest integer such that $2^\kappa \mid d$.

Proof. By Lemma 1, there are $2^{\kappa+\ell}$ distinct pairs (j, k) with argument α up to sign for α an integer on $0 \leq \alpha < 2^{m+\ell}$ such that $2^\kappa \mid \alpha$. Both negative and positive signs are admissible, except when α is zero as the sign then has no meaning, and when $\alpha = 2^{\ell+m-1}$, as only negative signs are then admissible.

This accounts for all pairs (j, k) where $0 \leq j < 2^{m+\ell}$ and $0 \leq k < 2^{\ell}$, as

$$\left(\frac{2^{\ell+m-1}}{2^{\kappa}} - 1 \right) \cdot 2^{\kappa+\ell+1} + 2 \cdot 2^{\kappa+\ell} = 2^{2\ell+m},$$

and so the lemma follows. ■

By Lemma 2, the number of pairs (j, k) having admissible arguments $\alpha = |\,\{dj + 2^m k\}_{2^{\ell+m}}\,|$ are uniformly distributed up to the sign of $\{dj + 2^m k\}_{2^{\ell+m}}$.

2.5 The probability of observing a pair with argument α

We have derived closed-form expressions for the number of pairs $N(\alpha)$ with argument α and for the probability $P(\theta(\alpha))$ of observing a specific pair (j, k) with angle θ, or equivalently, with argument α.

This immediately yields a closed-form expression for the probability $\Phi(\alpha)$ of observing some pair with argument α, as

$$\Phi(\alpha) = N(\alpha) \cdot P\left(\theta(\alpha)\right) \qquad \text{where} \qquad \theta(\alpha) = \frac{2\pi\alpha}{2^{\ell+m}}.$$

3 Modelling the probability density function

In this section, we sum $\Phi(\alpha)$ numerically over partial intervals in α to form a high resolution histogram for $\Phi(\alpha)$, and we demonstrate analytically that the probability of observing a pair (j, k) with argument $\alpha \ggg 2^m$ is negligible

3.1 The notion of t-good pairs

To partition the pairs, we propose to introduce the notion of a t-good pair.

Definition 1. *A pair (j, k) is said to be t-good if its argument is on the interval*

$$2^{t-1} \leq \alpha = \left|\, \{dj + 2^m k\}_{2^{\ell+m}} \,\right| < 2^t$$

for t an integer on the interval $0 < t \leq \ell + m$. For t equal to zero, a pair is said to be t-good if its argument is zero.

3.2 The probability $\rho(t)$ of observing a t-good pair

The probability of observing a t-good pair is

$$\rho(t) = \sum_{\alpha} \Phi(\alpha)$$

where the sum is over all α on $2^{t-1} \leq \alpha < 2^t$ for $0 < t \leq \ell + m$ whilst at the two endpoints $\rho(0) = \Phi(0)$ and $\rho(\ell + m) = \Phi(2^{\ell+m-1})$.

It is natural to partition the pairs in this manner, as $\Phi(\alpha)$ has a very long tail, and as the approximate size of the argument of a pair determines how easy it is to recover d from the pair. Therefore, we are interested in understanding roughly how the probability mass of $\Phi(\alpha)$ is distributed with respect to the size of the argument, and $\rho(t)$ provides this information.

3.3 Bounding the probability $\rho(t)$

In this section, we prove that the probability of observing a pair (j, k) with argument significantly greater than 2^m is negligible.

Lemma 3. *The probability $\rho(t)$ is upper bounded by $\frac{20}{\pi^2} \cdot 2^{m-t}$.*

Proof. For $t \leq m$ the lemma follows trivially. For $t > m$, recall that

$$\rho(t) = \sum_{\alpha} \Phi(\alpha) = \frac{1}{2^{2(m+2\ell)}} \sum_{\alpha} N(\alpha) \sum_{e} \xi(\theta, \#b(e)) \quad \text{where} \quad \theta = \frac{2\pi\alpha}{2^{\ell+m}}.$$

The first sum is over the $2^{t-\kappa-1}$ arguments α on $2^{t-1} \leq \alpha < 2^t$ such that $2^\kappa \mid \alpha$. There are $N(\alpha) \leq 2^{\ell+\kappa+1}$ pairs with argument α. The second sum is over at most $2^{\ell+m+1}$ values of e. Furthermore, as $1 - \cos(\theta \cdot \#b(e)) \leq 2$, and as $1 - \cos\theta \geq \theta^2/5$ for $|\,\theta\,| \leq \pi$, we may bound

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

Combining these bounds and using that $\theta = 2\pi\alpha/2^{\ell+m} \geq 2^t\pi/2^{\ell+m}$ yields

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

and so the lemma follows. ■

One implication of Lemma 3 is that even if the quantum algorithm does not yield a good pair by the definition in [3], it will with great probability yield a pair that is close to being good. This indicates that it should be possible to solve for d even if not all pairs included in the lattice L are good.

3.4 Numerical approximations of $\Phi(\alpha)$ and $\rho(t)$

In this section, we proceed to use numerical methods to approximate $\rho(t)$.

More specifically, we use algorithm 1 in appendix A that simply selects a subset of equidistant elements from the sum in the expression for $\rho(t)$ and interpolates between them using the rectangle method, the trapezoid method and Newton-Coates' method to approximate the function. Inadmissible arguments known to yield a zero contribution are excluded from the selection.

By varying the number of elements included in the selection it is possible to make a tradeoff between the time complexity of the algorithm and the accuracy of the approximation. If all elements are included in the selection, the algorithm yields an exact result. The maximum number of elements included is upper bounded by 2^ν where ν is a parameter to the algorithm.

The step size, that is the distance between the elements selected, is set to the least multiple of 2^κ to respect the upper bound of 2^ν on the number of elements selected in total. If the step size is 2^κ then all arguments that yield a non-zero contribution to $\rho(t)$ are included in the selection. An exact result is computed and there is no need to interpolate. If the step size is $2 \cdot 2^\kappa$ interpolation is performed using the trapezoid method. For greater multiples of 2^κ, interpolation is performed using Newton-Coates' method.

Algorithm 1 returns a list, the entries of which consist of a set of arguments and the probability of observing one of these arguments. The actual sum $\rho(t)$ is computed in algorithm 3. The reason for this subdivision is that algorithm 1 is used as a subroutine in other algorithms in Sect. 4.

3.5 Results and analysis

Histograms for $\rho(t)$ as approximated numerically by algorithms 1 and 3 are drawn in Fig. 2, for parameters $m = \ell = 256$ and $\nu = 6$, and for approximately equidistant values of d on the interval $2^{m-1} \leq d < 2^m$ as well as for the case where d is smaller than 2^m by some orders of magnitude.

As is evident from these histograms, the size of d in relation to 2^m affects $\rho(t)$, as does the greatest power of two 2^κ to divide d. The values of d have been selected to highlight the various extreme cases. The cases where κ is close to its maximum are rather esoteric, as d is trivial to compute classically when divisible by an almost maximal power of two. They are included for completeness.

The histograms drawn in Fig. 2 are representative of most choices of the parameters m and $\ell = m/s$, provided m and/or ℓ are not extremely small.

All histograms are drawn with resolution $\nu = 6$. Further increasing ν only yields a marginally better approximation, at the expense of an exponential cost increase when computing the approximation. We shall therefore fixate $\nu = 6$ throughout the remainder of this paper.

In the right hand part of Fig. 2 in the dashed region denoted A, all d are selected odd so κ is zero. As d decreases from $2^m - 1$ to $2^{m-1} + 1$, the probability mass is shifted towards slightly lesser values of t. However, the effect is moderate. This re-distribution is expected; it reflects the fact that it becomes eaiser to solve for d as d decreases in size in relation to 2^m.

In the lower left hand corner of Fig. 2 in the dashed region denoted B, an additional histogram is drawn for $d = 2^{m-5} - 1$. This histogram is representative of the distribution that arises when d is smaller than 2^m by at least a few orders of magnitude. Further decreasing d has no significant effect on the distribution.

In the upper left hand part of Fig. 2 in the region denoted C, the values of d are chosen to be divisible by great powers of two. This implies that κ is close to its maximum. Since $N(\alpha)$ is zero for all non-zero arguments α not divisible by 2^κ, all entries in the histogram on the interval $0 < t < \kappa$ are forced to zero and the probability mass is re-distributed accordingly. This effect arises for all non-zero κ. However, it is negligible when κ is not close to its maximum.

Based upon the above analysis, we conjecture the probability p of observing a good pair to, in general, be considerably greater than the lower bound of $p \geq 1/8$ guarantees, see [2,3] and Sect. 1.4. In general, it should hence be safe to assume a lower bound of $p \geq 3/10$ assuming random d on $2^{m-1} \leq d < 2^m$.

As was described in Sect. 1.4, all subsets of s pairs from a set of s/p pairs resulting from s/p quantum algorithm runs need to be exhaustively enumerated to compute d using the post-processing algorithm originally proposed. Hence, an increased expected success probability p implies complexity reductions, both in terms of the expected number of runs of the quantum algorithm required, and in terms of the number of lattices that need to be exhaustively enumerated.

4 Simulating the quantum algorithm

In this section, we describe how the high resolution histogram for $\Phi(\alpha)$ may be used to classically simulate the quantum algorithm when d is known.

The idea is simply to regard the argument α as a stochastic variable with probability density function $\Phi(\alpha)$ and to use the histogram for $\Phi(\alpha)$ produced

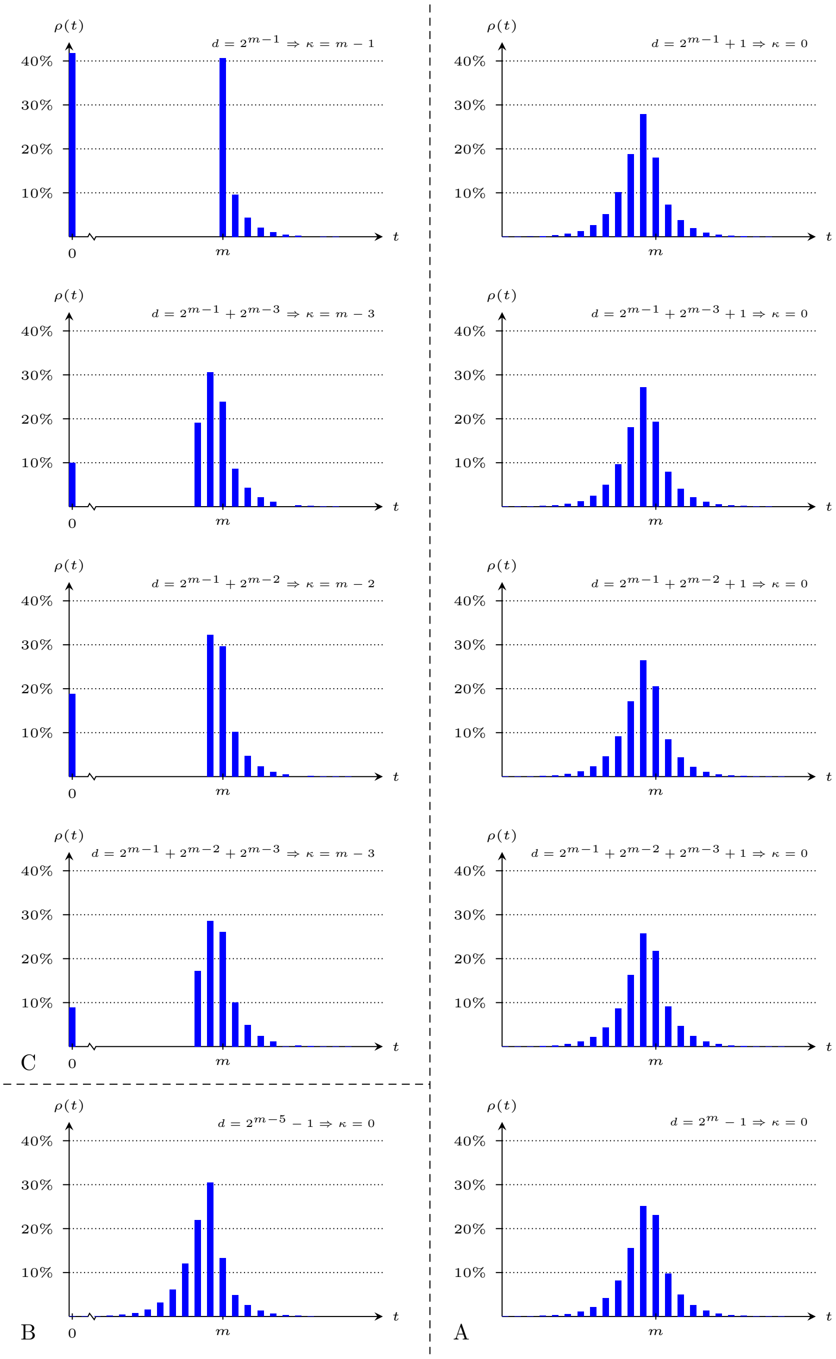

Figure 2: Plots of $\rho(t)$ for $m = \ell = 256$ and $\nu = 6$. Essentially the same situation arises for essentially any permissible combination of m and ℓ, and for any $\nu \geq 6$.

by algorithm 2 in appendix A to sample the variable. More specifically, a bin in the histogram is first selected according to the probability of an argument in it being observed as given by $\Phi(\alpha)$. The argument is then selected uniformly at random from the set of all arguments in this bin. For details, see algorithm 4.

The first step is self-explanatory. Our rationale for using uniform sampling in the second step is that the probability of observing a pair with a given argument is determined solely by the size of the argument. All arguments in the set will be of approximately the same size assuming high resolution in the histogram.

Given an argument α sampled by algorithm 4, a pair (j, k) with argument α is selected uniformly at random from the set of all pairs with argument α by algorithm 5 in appendix A. A proof of its correctness is given in Lemma 1 and 2. We conjecture pairs (j, k) sampled in this manner, see the combined procedure in algorithm 6 in appendix A, to be representative of pairs produced by the quantum algorithm, enabling us to classically simulate the quantum algorithm when d is known.

5 Verifying the original post-processing

In this section, we take the opportunity to verify the post-processing algorithm originally proposed in [2, 3] by simulating the quantum algorithm.

More specifically, for all pairwise combinations of $m \in \{128, 256, \ldots, 8192\}$ and $s \in \{1, 2, \ldots, 8, 10, 12, 15, 20\}$, and for both maximal $d = 2^{m-1}$ and random d on $2^{m-1} \leq d < 2^m$, we fixate $\ell = \lceil m/s \rceil$ and $\nu = 6$, and sample 10^2 sets of s good pairs (j, k) by the definition in [3], see algorithm 7 in appendix A. Each set is then solved for d using the original post-processing algorithm. Executing this procedure produced the results in the next section.

The enumeration algorithm employed is described in appendix C. The basis was reduced using the block Korkin-Zolotarev (BKZ) [5, 8] algorithm, as it is implemented in fpLLL provided by Sage v7.2, with default parameters and a block size of ten for all combinations of m and s. For the combinations of m and s considered, a lattice basis takes at most minutes to reduce and solve for d in a single execution thread on an ordinary workstation.

We remark that sampling all pairs to be good avoids the need to exhaustively solve subsets of pairs for d, enabling us to verify the post-processing algorithm for greater values of s than would otherwise be practical.

5.1 Results and analysis

For maximal $d = 2^m - 1$, the classical post-processing algorithm succeed in recovering d from all 10^2 sets. The lattice enumeration algorithm completed very quickly, after exploring at most a few vectors in the lattice.

For random d on $2^{m-1} \leq d < 2^m$, the post-processing algorithm succeeded in recovering d for all 10^2. As expected, fewer vectors needed to be enumerated when d is random compare to when d is maximal.

For combinations of m and s where s does not divide m so that ℓ is rounded upwards, it is often sufficient to map $\mathbf{v}$ to the closest vector in L to find $\mathbf{u}$. This situation arises more frequently for random d than for maximal d.

6 Our improved post-processing algorithm

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

Note that the pairs are not required to be good by the definition in [3]. The algorithm therefore does not require an exhaustive search over subsets of pairs to be performed. It furthermore does not require lattice enumeration provided n is selected sufficiently large in relation to m and s.

6.1 Recovering d from a set of n pairs

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 with probability at least q the distance

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

assuming $\tau(n)$ is selected so that

$$\Pr\left[\frac{1}{n} \sum_{i=1}^{n} \alpha_i^2 \leq 2^{2(m+\tau(n))} \right] \geq q \qquad (2)$$

where α_i is a stochastic variable with probability density function $\Phi(\alpha_i)$.

The volume of the fundamental parallelepiped of L is $|\det L\,| = 2^{(\ell+m)n}$ and this volume contains a single lattice point. On the other hand, the volume of a D-dimensional hypersphere of radius R in Euclidean space is

$$V_D(R) = \frac{\pi^{D/2}}{\Gamma\left(\frac{D}{2} + 1\right)} R^D$$

where Γ is the gamma function.

We expect to find $\mathbf{u}$ simply by mapping $\mathbf{v}$ to the closest vector in L when the volume quotient $v = V_D(R) \, / \, | \det L \,| < 2$ since the hypersphere is then expected to contain a single lattice vector. Hence, if n is selected such that

$$v = \frac{V_D(R)}{|\det L\,|} = \frac{\pi^{(n+1)/2}}{\Gamma\left(\frac{n+1}{2}+1\right)} \left(2^m \sqrt{2^{2\tau(n)}\, n + 1}\right)^{n+1} \cdot \frac{1}{2^{(m+\ell)n}} < 2,$$

with probability q for some constant $\tau(n)$, then it should be possible, again with probability q, to recover $\mathbf{u}$ from $\mathbf{v}$ by mapping $\mathbf{v}$ to the closest vector in L.

In practice, the closest vector is found by reducing the basis using a lattice basis reduction algorithm and applying Babai's [1] nearest plane algorithm. To perform the reduction algorithms ranging from e.g. Lenstra-Lenstra-Lovász [6] to Korkin-Zolotarev [5, 8] may be employed. The more strongly reduced the basis, the smaller the error will be when applying Babai's algorithm.

6.2 Estimating $\tau(n)$

To estimate $\tau(n)$ for parameters m and s, and some fixated probability q, we sample N sets of n arguments $\{\alpha_1, \ldots, \alpha_n\}$. For each set, we compute τ from

$$\frac{1}{n}\sum_{i=1}^{n} \alpha_i^2 = 2^{2(m+\tau)} \quad \Rightarrow \quad \tau = \frac{1}{2}\log_2 \frac{1}{n}\sum_{i=1}^{n}\alpha_i^2 - m,$$

sort the resulting set of values of τ in increasing order, and select the value at index $\lceil (N-1)q \rfloor$ in the resulting list to arrive at the estimate for $\tau(n)$.

The integer 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 a sufficiently good estimate of $\tau(n)$.

6.3 Results and analysis

To estimate n as a function of m and s, and to verify the estimates in simulations, we fixate $q = 0.99$ and $\nu = 6$, and consider the hardest case $d = 2^m - 1$.

For relevant combinations of m and s, we let $\ell = \lceil m/s \rceil$, fixate $N = 10^6$ when estimating $\tau(n)$, 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.

Table 1 was produced by executing these procedures. For detailed results, see tables 2 – 8 in appendix B. To reduce the lattice bases, the block Korkin-Zolotarev (BKZ) algorithm [5, 8] was employed, as it is implemented in fpLLL provided by Sage v7.2, with default parameters and a block size of ten for all combinations of m, s and n. For the combinations of m, s and n considered, a lattice basis takes at most minutes to reduce and solve for d in a single execution thread on an ordinary workstation.

The estimated values of n are verified by the simulations, except when v is close to but less than two, requiring n to be incremented in the simulations. The entries in the table for which this is the case are printed in bold.

		logarithm length m						
		128	256	512	1024	2048	4096	8192
tradeoff factor s	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	8	6	6	6	6	6	6
	6	10	8	7	7	7	7	7
	7	13	**9**	8	8	8	8	8
	8	18	**11**	10	9	9	9	9
	10	32	**15**	12	11	11	11	11
	20	–	71	**31**	24	22	21	21
	30	–	–	59	40	35	33	31
	40	–	–	–	62	**48**	44	42
	50	–	–	–	–	66	57	**53**

		logarithm length m						
		128	256	512	1024	2048	4096	8192
tradeoff factor s	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	8	6	6	6	6	6	6
	6	10	8	7	7	7	7	7
	7	13	**10**	8	8	8	8	8
	8	18	**12**	10	9	9	9	9
	10	–	**16**	12	11	11	11	11
	20	–	–	**32**	24	22	21	21
	30	–	–	–	40	35	33	31
	40	–	–	–	–	**49**	44	42
	50	–	–	–	–	–	57	**54**

Table 1: The estimated (top) and simulated (bottom) 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 without enumerating L with probability $\approx 99\%$.

On the lower left diagonal, the variance in α is too great, and the decrements in v with respect to n too small, for reliable data to be produced without increasing M and N. This explains the lack of simulations for these estimates.

Compared to the post-processing algorithm originally proposed, the new post-processing algorithm achieves considerably better tradeoffs with practical time complexity for cryptographically relevant parameter choices. In general, it requires considerably fewer quantum algorithm runs.

Further increasing s is possible in practice and would yield slightly better tradeoffs. However, it is likely not worthwhile as the improvement would come at the expense of having to run the quantum algorithm many more times.

The parameter choices made above are conservative, in that we have fixated q at 99% and considered the hardest case of $d = 2^m - 1$. Furthermore, we have made a conservative choice in requiring that mapping $\mathbf{v}$ to the closest vector in L should immediately produce $\mathbf{u}$ without enumerating vectors in L.

In practice, some of these choices may be relaxed: The logarithm d would typically be randomly selected on $2^{m-1} \leq d < 2^m$. It is possible to select m slightly greater than the bit length of d. Furthermore, the technique of increasing the lattice dimension may be combined with lattice enumeration and subset inclusion or exclusion techniques as in the orignal post-processing algorithm.

7 Summary and conclusion

Our main contribution in this paper has been 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 algorithm.

Enumerating exponential in s many $s+1$-dimensional lattices, as in the post-processing algorithm originally proposed [3], may thus be avoided at the expense of reducing a single $n+1$-dimensional lattice. By simulating the quantum algorithm for known d we have estimated n as a function of m and s. We have verified our estimates by post-processing simulated outputs.

The new post-processing algorithm is practical for larger tradeoff factors s and, in general, requires fewer quantum algorithm runs compared to the original post-processing algorithm. The new algorithm enables good tradeoffs to be achieved in practice for cryptographically relevant parameters choices.

These results are relevant for RSA and for cryptographic schemes based on the computational intractability of the short discrete logarithm problem such as Diffie-Hellman when instantiated with safe-prime groups and short exponents.

Acknowledgments

Many thanks to Johan Håstad for valuable comments and guidance. Thanks to Lennart Brynielsson and other colleagues for useful discussions.

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

References

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

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

[3] Ekerå, M., Håstad, J.: Quantum algorithms for computing short discrete logarithms and factoring RSA integers. In: Lange T., Takagi T. (Eds) Post-Quantum Cryptography. PQCrypto 2017. LNCS, vol. 10346, pp. 347–363. Springer, Cham (2017)

[4] Kannan, R.: Improved algorithms for integer programming and related lattice problems. In: Proceedings of the 15th Symposium on the Theory of Computing. STOC 1983. ACM Press, pp. 99—108 (1983).

[5] Korkine, A., Zolotareff, G.: Sur les formes quadratiques. Math. Ann. **6**(3), pp. 366–389 (1873)

[6] Lenstra, H.W., Lenstra, A.K., Lovász, L.: Factoring Polynomials with Rational Coefficients. Math. Ann. **261**(4) pp. 515–534 (1982)

[7] 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).

[8] Schnorr, C.P.: A hierarchy of polynomial time lattice basis reduction algorithms. Theoretical Computer Science, **53**(2–3), pp. 201–224 (1987)

[9] 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)

[10] 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)

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

A Algorithms

In this appendix, we specify algorithms for numerically approximating $\rho(t)$ and $\Phi(\alpha)$ and for sampling arguments and pairs.

A.1 Approximating $\Phi(\alpha)$ and $\rho(t)$

Algorithm 1 Computes a partial histogram for $\Phi(\alpha)$.

```
procedure PartialHistogramPhi($t$, $\ell$, $m$, $\nu$, $d$)
    Let $\kappa$ be such that $2^\kappa$ is the greatest power of two to divide $d$.
    if $t = 0$ then                                          ▷ Single entry intervals.
        return $[\,\{0\},\ \Phi(0)\,]$.
    else if $t = m + l$ then
        return $[\,\{2^{\ell+m-1}\},\ \Phi(2^{\ell+m-1})\,]$.
    end if
    Let $s = \max(2^\kappa, 2^{t-1}/2^\nu)$.        ▷ The step size $s \geq 2^\kappa$ controlled by $\nu$.
    Let $\mathcal{H} = [\,]$ be an empty ordered list.
    if $s = 2^\kappa$ then                                   ▷ Rectangle method.
        for $\alpha \in \{2^{t-1},\ 2^{t-1} + s,\ \ldots,\ 2^t - s\}$ do
            Append $[\,\{\alpha\},\ \Phi(\alpha)\,]$ to $\mathcal{H}$.
        end for
    else if $s = 2 \cdot 2^\kappa$ then                                        ▷ Trapezoid method.
        for $\alpha \in \{2^{t-1},\, 2^{t-1} + s,\, \ldots,\, 2^t - s\}$ do
            Let $\varsigma = s/2^\kappa \cdot (\Phi(\alpha) + \Phi(\alpha + s))/2 = \Phi(\alpha) + \Phi(\alpha + s)$.
            Append $[\,\{\alpha,\, \alpha + 2^\kappa\},\, \varsigma\,]$ to $\mathcal{H}$.
        end for
    else                                                                   ▷ Newton-Cotes' method.
        for $\alpha \in \{2^{t-1},\, 2^{t-1} + s,\, \ldots,\, 2^t - s\}$ do
            Let $\varsigma = s/2^\kappa \cdot (\Phi(\alpha) + 4\Phi(\alpha + s/2) + \Phi(\alpha + s))/6$.
            Append $[\,\{\alpha,\, \alpha + 2^\kappa,\, \ldots,\, \alpha + s - 2^\kappa\},\, \varsigma\,]$ to $\mathcal{H}$.
        end for
    end if
    return $\mathcal{H}$.
end procedure
```

Algorithm 2 Computes a complete histogram for $\Phi(\alpha)$.

```
procedure HistogramPhi($\ell$, $m$, $\nu$, $d$)
    Let $\mathcal{H} = [\,]$ be an empty ordered list.
    for $t \in \{0,\, 1,\, \ldots,\, \ell + m\}$ do
        Concatenate PartialHistogramPhi($t, \ell, m, \nu, d$) to $\mathcal{H}$.
    end for
    return $\mathcal{H}$.
end procedure
```

Algorithm 3 Computes the probability $\rho(t)$ of observing a t-good pair

```
procedure Rho($t$, $\ell$, $m$, $\nu$, $d$)
    Let $\mathcal{H}$ = PartialHistogramPhi($t, \ell, m, \nu, d$).
    Let $\rho(t) = 0$.
    for $[\,\mathcal{S},\, \varsigma\,]$ in $\mathcal{H}$ do
        Add $\varsigma$ to $\rho(t)$.
    end for
    return $\rho(t)$.
end procedure
```

A.2 Sampling arguments and pairs

Algorithm 4 Samples an argument α according to $\Phi(\alpha)$.

```
procedure SampleArgument($\ell$, $m$, $\nu$, $d$)
    Let $\mathcal{H}$ = HistogramPhi($\ell, m, \nu, d$).
    Select $\lambda$ uniformly at random on $0 \le \lambda < 1$.
    for $[\,\mathcal{S},\, \varsigma\,]$ in $\mathcal{H}$ do
        Subtract $\varsigma$ from $\lambda$.
        if $\lambda \le 0$ then
            return $\alpha$ selected uniformly at random from the set $\mathcal{S}$.
        end if
    end for
```

```
  end procedure
```

Algorithm 5 Samples a pair (j,k) with argument α uniformly at random.

```
  procedure SamplePairWithArgument($\alpha$, $\ell$, $m$, $d$)
      Let $\kappa$ be such that $2^\kappa$ is the greatest power of two to divide $d$.
      if $\alpha = 0$ then
          Let $s = 1$.                                   ▷ The sign is of no consequence.
      else if $\alpha = 2^{\ell+m-1}$ then
          Let $s = -1$.                                  ▷ The sign must be negative.
      else
          Select $s$ uniformly at random from $\{1, -1\}$.     ▷ Pick a random sign.
      end if
      Select $r$ uniformly at random on $0 \leq r < 2^\kappa$.   ▷ Pick one of $2^\kappa$ values $r$.
      Select $k$ uniformly at random on $0 \leq k < 2^\ell$.     ▷ Pick a random $k$.
      Let $j = \left( \frac{s\alpha - 2^m k}{2^\kappa} \left(\frac{d}{2^\kappa}\right)^{-1} + 2^{\ell+m-\kappa} r \right) \bmod 2^{\ell+m}$.   ▷ Solve for $j$.
      return $(j,k)$.
  end procedure
```

Algorithm 6 Samples a pair (j,k) with argument α according to $\Phi(\alpha)$.

```
  procedure SamplePair($\ell$, $m$, $\nu$, $d$)
      Let $\alpha$ = SampleArgument($\ell, m, \nu, d$).
      return $(j,k)$ where $(j,k)$ = SamplePairWithArgument($\alpha, \ell, m, d$).
  end procedure
```

Algorithm 7 Samples a good pair (j,k) with argument α according to $\Phi(\alpha)$.

```
  procedure SampleGoodPair($\ell$, $m$, $\nu$, $d$)
      loop
          Let $\alpha$ = SampleArgument($\ell, m, \nu, d$).
          if $\alpha \leq 2^{m-2}$ then        ▷ Condition from [3] on the pair being good.
              break the loop
          end if
      end loop
      return $(j,k)$ where $(j,k)$ = SamplePairWithArgument($\alpha, \ell, m, d$).
  end procedure
```

B Tabulated results

In this appendix, we tabulate results from the estimates and simulations of the new post-processing algorithm described in Sect. 6. For each value of the logarithm length m, we tabulate the tradeoff factor s, the number of pairs n, the estimated value of $\tau(n)$, the volume quotient v and the error when simulating the quantum algorithm and executing classical post-processing.

The error indicates the number of times that reducing the lattice basis and applying Babai's algorithm failed to produce $\mathbf{u}$ and hence d without enumerating when performing $M = 10^3$ simulations. Note that the variance in the error is non-negligible for some choices of m, s and n. The error column should only be regarded as providing an approximate indication of the size of the error.

m	s	n	$\tau(n)$	v	error (‰)
128	**1**	**2**	4.84	$8.12 \cdot 10^{-34}$	0
128	**2**	**3**	5.13	$3.63 \cdot 10^{-12}$	0
128	**3**	**4**	5.36	$1.13 \cdot 10^{-3}$	2
128	4	5	5.49	$1.25 \cdot 10^{3}$	48
128	**4**	**6**	5.62	$9.62 \cdot 10^{-5}$	7
128	5	6	5.61	$6.28 \cdot 10^{6}$	–
128	5	7	5.72	$3.17 \cdot 10^{1}$	15
128	**5**	**8**	5.81	$1.79 \cdot 10^{-4}$	6
128	6	8	5.86	$1.02 \cdot 10^{6}$	–
128	6	9	5.94	$9.73 \cdot 10$	16
128	**6**	**10**	6.00	$8.57 \cdot 10^{-3}$	5
128	7	10	5.98	$8.40 \cdot 10^{6}$	–
128	7	11	6.09	$9.61 \cdot 10^{3}$	–
128	7	12	6.14	7.59	13
128	**7**	**13**	6.19	$7.06 \cdot 10^{-3}$	7
128	8	13	6.17	$3.20 \cdot 10^{9}$	–
128	8	14	6.27	$3.78 \cdot 10^{7}$	–
⋮	⋮	⋮	⋮	⋮	⋮
128	8	17	6.36	$1.30 \cdot 10$	12
128	**8**	**18**	6.43	$1.59 \cdot 10^{-1}$	8
128	10	18	6.43	$2.83 \cdot 10^{15}$	–
128	10	19	6.43	$1.22 \cdot 10^{14}$	–
⋮	⋮	⋮	⋮	⋮	⋮
128	10	31	6.79	$1.36 \cdot 10^{1}$	11
128	**10**	**32**	6.79	$7.06 \cdot 10^{-1}$	10

Table 2: Results for $m = 128$.

m	s	n	$\tau(n)$	v	error (‰)
256	**1**	**2**	4.85	$2.48 \cdot 10^{-72}$	0
256	**2**	**3**	5.11	$1.87 \cdot 10^{-31}$	0
256	**3**	**4**	5.32	$5.50 \cdot 10^{-17}$	0
256	**4**	**5**	5.50	$3.01 \cdot 10^{-7}$	1
256	**5**	**6**	5.64	$2.72 \cdot 10^{-2}$	1
256	6	7	5.75	$1.93 \cdot 10^{4}$	40
256	**6**	**8**	5.82	$7.19 \cdot 10^{-7}$	1
256	7	8	5.82	$2.08 \cdot 10^{8}$	–
256	**7**	**9**	5.92	$6.42 \cdot 10^{-1}$	14
256	7	10	5.99	$2.47 \cdot 10^{24}$	2
256	8	10	6.00	$2.32 \cdot 10^{6}$	–

m	s	n	$\tau(n)$	v	error (‰)
256	**8**	**11**	6.11	$3.63 \cdot 10^{-1}$	13
256	8	12	6.14	$3.09 \cdot 10^{-8}$	5
256	10	12	6.09	$9.19 \cdot 10^{13}$	–
256	10	13	6.19	$9.86 \cdot 10^{8}$	–
256	10	14	6.26	$8.56 \cdot 10^{3}$	–
256	**10**	**15**	6.28	$4.95 \cdot 10^{-2}$	12
256	10	16	6.35	$5.09 \cdot 10^{-7}$	6
256	20	21	6.09	$9.19 \cdot 10^{13}$	–
256	20	22	6.19	$9.86 \cdot 10^{8}$	–
⋮	⋮	⋮	⋮	⋮	⋮
256	20	70	7.30	$2.64 \cdot 10$	–
256	**20**	**71**	7.28	$9.63 \cdot 10^{-1}$	–

Table 3: Results for $m = 256$.

m	s	n	$\tau(n)$	v	error (‰)
512	**1**	**2**	4.85	$2.12 \cdot 10^{-149}$	0
512	**2**	**3**	5.14	$5.94 \cdot 10^{-70}$	0
512	**3**	**4**	5.33	$2.95 \cdot 10^{-42}$	0
512	**4**	**5**	5.49	$1.57 \cdot 10^{-26}$	0
512	**5**	**6**	5.65	$2.45 \cdot 10^{-17}$	0
512	**6**	**7**	5.76	$5.71 \cdot 10^{-10}$	1
512	**7**	**8**	5.84	$2.15 \cdot 10^{-4}$	7
512	8	9	5.90	$4.77 \cdot 10^{3}$	22
512	**8**	**10**	5.99	$1.17 \cdot 10^{-13}$	0
512	10	11	6.07	$1.77 \cdot 10^{10}$	89
512	**10**	**12**	6.14	$1.99 \cdot 10^{-3}$	8
512	20	21	6.55	$3.81 \cdot 10^{45}$	–
512	20	22	6.57	$2.85 \cdot 10^{40}$	–
⋮	⋮	⋮	⋮	⋮	⋮
512	20	30	6.79	3.51	–
512	**20**	**31**	6.86	$1.09 \cdot 10^{-4}$	11
512	20	32	6.84	$5.34 \cdot 10^{-10}$	10
512	30	31	6.83	$2.62 \cdot 10^{70}$	–
512	30	32	6.86	$9.68 \cdot 10^{67}$	–
⋮	⋮	⋮	⋮	⋮	⋮
512	**30**	**59**	7.26	$1.59 \cdot 10$	–
512	30	60	7.28	$9.60 \cdot 10^{-2}$	–

Table 4: Results for $m = 512$.

m	s	n	$\tau(n)$	v	error (‰)
1024	**1**	**2**	4.81	$1.46 \cdot 10^{-303}$	0

m	s	n	$\tau(n)$	v	error (‰)
1024	**2**	**3**	5.12	$4.79 \cdot 10^{-147}$	0
1024	**3**	**4**	5.35	$5.27 \cdot 10^{-94}$	0
1024	**4**	**5**	5.49	$4.51 \cdot 10^{-65}$	0
1024	**5**	**6**	5.63	$1.82 \cdot 10^{-47}$	0
1024	**6**	**7**	5.73	$5.18 \cdot 10^{-35}$	0
1024	**7**	**8**	5.83	$4.23 \cdot 10^{-26}$	0
1024	**8**	**9**	5.93	$3.15 \cdot 10^{-16}$	0
1024	**10**	**11**	6.08	$3.23 \cdot 10^{-5}$	7
1024	20	21	6.57	$2.75 \cdot 10^{35}$	1
1024	20	22	6.57	$2.38 \cdot 10^{22}$	–
1024	20	23	6.61	$3.90 \cdot 10^{9}$	38
1024	**20**	**24**	6.62	$4.12 \cdot 10^{-4}$	4
1024	30	31	6.79	$3.14 \cdot 10^{65}$	–
1024	30	32	6.83	$1.21 \cdot 10^{58}$	–
⋮	⋮	⋮	⋮	⋮	⋮
1024	30	39	6.98	$5.72 \cdot 10^{4}$	20
1024	**30**	**40**	6.98	$9.99 \cdot 10^{-4}$	9
1024	40	41	7.01	$4.23 \cdot 10^{100}$	–
1024	40	42	7.05	$9.47 \cdot 10^{95}$	–
⋮	⋮	⋮	⋮	⋮	⋮
1024	40	61	7.30	$6.74 \cdot 10^{3}$	–
1024	**40**	**62**	7.31	$1.38 \cdot 10^{-1}$	–

Table 5: Results for $m = 1024$.

m	s	n	$\tau(n)$	v	error (‰)
2048	**1**	**2**	4.86	$8.96 \cdot 10^{-612}$	0
2048	**2**	**3**	5.13	$3.74 \cdot 10^{-301}$	0
2048	**3**	**4**	5.35	$2.41 \cdot 10^{-196}$	0
2048	**4**	**5**	5.49	$3.97 \cdot 10^{-142}$	0
2048	**5**	**6**	5.62	$1.62 \cdot 10^{-109}$	0
2048	**6**	**7**	5.75	$4.77 \cdot 10^{-87}$	0
2048	**7**	**8**	5.85	$2.18 \cdot 10^{-69}$	0
2048	**8**	**9**	5.92	$8.61 \cdot 10^{-55}$	0
2048	**10**	**11**	6.08	$1.02 \cdot 10^{-34}$	0
2048	20	21	6.55	$1.50 \cdot 10^{21}$	101
2048	**20**	**22**	6.59	$1.06 \cdot 10^{-7}$	9
2048	30	31	6.83	$7.21 \cdot 10^{56}$	–
2048	30	32	6.86	$1.12 \cdot 10^{39}$	–
⋮	⋮	⋮	⋮	⋮	⋮
2048	30	34	6.87	$1.13 \cdot 10^{3}$	22
2048	**30**	**35**	6.89	$1.27 \cdot 10^{-15}$	6
2048	40	41	6.98	$3.76 \cdot 10^{87}$	–
2048	40	42	7.03	$1.98 \cdot 10^{75}$	–

m	s	n	$\tau(n)$	v	error (‰)
⋮	⋮	⋮	⋮	⋮	⋮
2048	40	47	7.10	$4.30 \cdot 10^{11}$	–
2048	**40**	**48**	7.12	$1.20 \cdot 10^{-1}$	19
2048	40	49	7.14	$2.70 \cdot 10^{-14}$	8
2048	50	51	7.16	$6.30 \cdot 10^{129}$	–
2048	50	52	7.19	$5.44 \cdot 10^{120}$	–
⋮	⋮	⋮	⋮	⋮	⋮
2048	50	64	7.33	$4.56 \cdot 10^{8}$	–
2048	**50**	**65**	7.36	$5.02 \cdot 10^{-1}$	–

Table 6: Results for $m = 2048$.

m	s	n	$\tau(n)$	v	error (‰)
4096	**1**	**2**	4.83	$2.59 \cdot 10^{-1228}$	0
4096	**2**	**3**	5.15	$2.19 \cdot 10^{-609}$	0
4096	**3**	**4**	5.32	$2.67 \cdot 10^{-402}$	0
4096	**4**	**5**	5.51	$3.24 \cdot 10^{-296}$	0
4096	**5**	**6**	5.65	$1.78 \cdot 10^{-233}$	0
4096	**6**	**7**	5.75	$4.19 \cdot 10^{-189}$	0
4096	**7**	**8**	5.84	$1.58 \cdot 10^{-158}$	0
4096	**8**	**9**	5.91	$7.15 \cdot 10^{-132}$	0
4096	**10**	**11**	6.09	$5.50 \cdot 10^{-97}$	0
4096	**20**	**21**	6.52	$5.08 \cdot 10^{-8}$	8
4096	30	31	6.84	$9.07 \cdot 10^{38}$	–
4096	30	32	6.85	2.82	16
4096	**30**	**33**	6.85	$7.36 \cdot 10^{-39}$	3
4096	40	41	7.02	$1.18 \cdot 10^{75}$	–
4096	40	42	7.02	$6.36 \cdot 10^{46}$	–
4096	40	43	7.05	$8.92 \cdot 10^{18}$	47
4096	**40**	**44**	7.07	$1.03 \cdot 10^{-9}$	7
4096	50	51	7.15	$5.41 \cdot 10^{116}$	–
4096	50	52	7.20	$3.88 \cdot 10^{95}$	–
⋮	⋮	⋮	⋮	⋮	⋮
4096	50	56	7.22	$2.42 \cdot 10^{8}$	25
4096	**50**	**57**	7.27	$2.14 \cdot 10^{-13}$	6

Table 7: Results for $m = 4096$.

m	s	n	$\tau(n)$	v	error (‰)
8192	**1**	**2**	4.85	$2.60 \cdot 10^{-2461}$	0
8192	**2**	**3**	5.12	$6.25 \cdot 10^{-1226}$	0
8192	**3**	**4**	5.35	$7.29 \cdot 10^{-813}$	0
8192	**4**	**5**	5.52	$1.88 \cdot 10^{-604}$	0

m	s	n	$\tau(n)$	v	error (‰)
8192	**5**	**6**	5.63	$9.19 \cdot 10^{-480}$	0
8192	**6**	**7**	5.74	$2.57 \cdot 10^{-395}$	0
8192	**7**	**8**	5.81	$2.02 \cdot 10^{-334}$	0
8192	**8**	**9**	5.93	$6.14 \cdot 10^{-286}$	0
8192	**10**	**11**	6.07	$1.15 \cdot 10^{-221}$	0
8192	**20**	**21**	6.54	$7.97 \cdot 10^{-71}$	0
8192	**30**	**31**	6.81	$1.52 \cdot 10^{-7}$	9
8192	40	41	7.02	$1.66 \cdot 10^{49}$	200
8192	**40**	**42**	7.03	$2.48 \cdot 10^{-10}$	8
8192	50	51	7.19	$3.15 \cdot 10^{91}$	–
8192	50	52	7.17	$3.78 \cdot 10^{44}$	–
8192	**50**	**53**	7.20	$2.88 \cdot 10^{-2}$	13
8192	50	54	7.18	$3.37 \cdot 10^{-49}$	0

Table 8: Results for $m = 8192$.

C Lattice enumeration

In this appendix, we describe a basic algorithm, borrowed from the work of Kannan [4], for enumerating all vectors in an integer lattice L that lie within a D-dimensional hypersphere of radius R centered on $\mathbf{v} \in \mathbb{Z}^D$.

We remark that there are more efficient enumeration algorithms, see for instance the more recent work of Micciancio and Walter [7].

C.1 Computing a reduced basis

The first step in the enumeration algorithm is to compute a reduced basis $\mathbf{B}$ for the lattice L. Let the basis vectors $\{\mathbf{b}_1, \ldots, \mathbf{b}_D\}$ of $\mathbf{B}$ be row vectors. Then

$$\mathbf{B} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_{D-1} \\ \mathbf{b}_D \end{bmatrix} \in \mathbb{Z}^{D \times D}.$$

To reduce the basis, algorithms ranging from e.g. Lenstra-Lenstra-Lovász [6] to Korkin-Zolotarev [5, 8] may be used. For lattices of small to moderate dimension these algorithms are efficient even when the lattices have large entries as is the case in this paper.

C.2 Computing a Gram-Schmidt-orthogonalized basis

The second step is to compute a Gram-Schmidt-orthogonalized basis $\mathbf{G}$ for $\mathbf{B}$, and a lower triangular matrix $\mathbf{M}$, such that $\mathbf{B}$ decomposes into

$$\underbrace{\begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_{D-1} \\ \mathbf{b}_D \end{bmatrix}}_{=\mathbf{B}\in\mathbb{Z}^{D\times D}} = \underbrace{\begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ \mu_{2,1} & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mu_{D-1,1} & \mu_{D-1,2} & \cdots & 1 & 0 \\ \mu_{D,1} & \mu_{D,2} & \cdots & \mu_{D,D-1} & 1 \end{bmatrix}}_{=\mathbf{M}\in\mathbb{Q}^{D\times D}} \underbrace{\begin{bmatrix} \mathbf{b}_1^* = \mathbf{b}_1 \\ \mathbf{b}_2^* \\ \vdots \\ \mathbf{b}_{D-1}^* \\ \mathbf{b}_D^* \end{bmatrix}}_{=\mathbf{G}\in\mathbb{Q}^{D\times D}}$$

or equivalently

$$\mathbf{b}_i = \mathbf{b}_i^* + \sum_{j=1}^{i-1} \mu_{ij}\mathbf{b}_j^* \quad \Leftrightarrow \quad \mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{ij}\mathbf{b}_j^*$$

where the projection factor μ_{ij} is defined as

$$\mu_{ij} = \frac{\langle \mathbf{b}_i,\, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*,\, \mathbf{b}_j^* \rangle}.$$

Let $\mathbf{u} \in L$. Then

$$\mathbf{u} = \mathbf{c}_u\mathbf{B} = (u_1,\, \ldots,\, u_D)\,\mathbf{B} = \sum_{i=1}^{D} u_i\mathbf{b}_i.$$

Since basis vector $\mathbf{b}_i$ for $1 \leq i \leq D$ may be written

$$\mathbf{b}_i = \mathbf{b}_i^* + \sum_{j=1}^{i-1} \mu_{ij}\mathbf{b}_j^*$$

the above sum may be written

$$\mathbf{u} = \sum_{i=1}^{D} u_i\mathbf{b}_i = \sum_{i=1}^{D} u_i \left(\mathbf{b}_i^* + \sum_{j=1}^{i-1} \mu_{ij}\mathbf{b}_j^* \right) = \sum_{j=1}^{D} \left(u_j + \sum_{i=j+1}^{D} \mu_{ij}u_i \right) \mathbf{b}_j^*.$$

where the last sum is column-wise over the matrix instead of row-wise, allowing us to sum the coefficient of each basis vector separately. Since the basis vectors $\{\, \mathbf{b}_1,\, \ldots,\, \mathbf{b}_D^* \,\}$ are orthogonal, the square norm

$$|\, \mathbf{u} \,|^2 = \left| \sum_{j=1}^{D} \left(u_j + \sum_{i=j+1}^{D} \mu_{ij}u_i \right) \mathbf{b}_j^* \right|^2 = \sum_{j=1}^{D} \left(u_j + \sum_{i=j+1}^{D} \mu_{ij}u_i \right)^2 |\, \mathbf{b}_j^* \,|^2.$$

C.3 The projection π_k

Define the projection $\pi_k(\mathbf{u})$ of $\mathbf{u}$ onto $\{\mathbf{b}_k^*,\, \ldots,\, \mathbf{b}_D^*\}$ as

$$\pi_k(\mathbf{u}) = \sum_{j=k}^{D} \left(u_j + \sum_{i=j+1}^{D} \mu_{ij}u_i \right) \mathbf{b}_j^*.$$

It is then easy to see that

$$0 \le |\,\pi_D(\mathbf{u})\,| \le |\,\pi_{D-1}(\mathbf{u})\,| \le \ldots \le |\,\pi_1(\mathbf{u})\,| = |\,\mathbf{u}\,|$$

since the above is equivalent to

$$0 \le |\,\pi_D(\mathbf{u})\,|^2 \le |\,\pi_{D-1}(\mathbf{u})\,|^2 \le \ldots \le |\,\pi_1(\mathbf{u})\,|^2 = |\,\mathbf{u}\,|^2$$

which is equivalent to

$$\begin{aligned}
0 &\le \underbrace{u_D^2\,|\,\mathbf{b}_D^*\,|^2}_{=\pi_D(\mathbf{u})} \\
&\le \underbrace{u_D^2\,|\,\mathbf{b}_D^*\,|^2 + (u_{D-1} + \mu_{D,D-1}u_D)^2\,|\,\mathbf{b}_{D-1}^*\,|^2}_{=\pi_{D-1}(\mathbf{u})} \\
&\le \underbrace{\sum_{j=k}^{D}\left(u_j + \sum_{i=j+1}^{D}\mu_{ij}u_i\right)^2 |\,\mathbf{b}_j^*\,|^2}_{=\pi_k(\mathbf{u})\text{ for }k=D-2,\ldots,1} \le |\,\pi_1(\mathbf{u})\,|^2 = |\,\mathbf{u}\,|^2.
\end{aligned}$$

This implies that for each orthogonal component we add to the projection, the norm of the projection can only increase. This fact enables us to efficiently enumerate the lattice recursively using the projection π_k.

C.4 Recursive enumeration

To recursively enumerate all vectors in L within radius R of $\mathbf{v}$, we first solve

$$\mathbf{v} = \mathbf{c}_v\mathbf{B} = (v_1, \ldots, v_D)\mathbf{B}$$

for $v_1, \ldots, v_D$ where $\mathbf{c}_v \in \mathbb{Q}^D$ since $\mathbf{v}$ is typically not in L. This implies that $(\lceil v_1 \rfloor, \ldots, \lceil v_D \rfloor)\,\mathbf{B}$ is a vector in L close to $\mathbf{v}$.

For $k = D, D-1, \ldots, 1$ we then compute an initial guess

$$\hat{u}_k = \left\lceil v_k - \sum_{j\,=\,k+1}^{D} \mu_{ij}(u_j - v_j) \right\rfloor$$

for u_k and recursively explore smaller values of k for all values $u_k = \hat{u}_k, \hat{u}_k \pm 1, \hat{u}_k \pm 2, \ldots$ respecting the condition that the distance $|\,\pi_k((\mathbf{c}_u - \mathbf{c}_v)\,\mathbf{B})\,| < R$.

Note that if $u_k = \hat{u}_k + i$ does not respect the condition, then no larger values of u_k can respect the condition. The same rule applies for $u_k = \hat{u}_k - i$. Note furthermore that $\hat{u}_k$ is the best initial guess for $u_k \in \mathbb{Z}$ since we need

$$u_k + \sum_{j\,=\,k+1}^{D} \mu_{ij}u_j \approx v_k + \sum_{j\,=\,k+1}^{D} \mu_{ij}v_j$$

where $v_k, \ldots, v_D \in \mathbb{Q}$ and $u_{k+1}, \ldots, u_D \in \mathbb{Z}$ are constants in order to minimize the distance $\mathbf{u} - \mathbf{v} = (\mathbf{c}_v - \mathbf{c}_u)\,\mathbf{B}$. The complete recursive procedure is described in pseudocode in algorithm 8.

Algorithm 8 Enumerating all vectors $\mathbf{u} \in L$ within distance R of $\mathbf{v}$.

```
procedure Enumerate($\mathbf{B} = (\mathbf{b}_1,\, \ldots,\, \mathbf{b}_D\,)^T$, $\mathbf{v}$, $R$)
    Compute the Gram-Schmidt-orthogonalized basis $\mathbf{G} = (\mathbf{b}_1^*,\, \ldots,\, \mathbf{b}_D^*)$

        where $\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{ij} \mathbf{b}_j^*$ and $\mu_{ij} = \frac{\langle \mathbf{b}_i,\, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*,\, \mathbf{b}_j^* \rangle}$.

    Solve $\mathbf{v} = \mathbf{c}_v \mathbf{B}$ for $\mathbf{c}_v = (v_1,\, \ldots,\, v_D)$.

    procedure EnumerateInner($\mathbf{c}_u = (u_1,\, \ldots,\, u_D)$, $k$)
        if $k = 0$ then
            return the set $\{\mathbf{u} = \mathbf{c}_u\, \mathbf{B}\}$.            ▷ Leaf node reached.
        end if

        Let $u_k = \hat{u}_k$ where $\hat{u}_k$ is defined as

            $\hat{u}_k = \left\lceil v_k - \sum_{j\,=\,k+1}^{D} \mu_{ij}(u_j - v_j) \right\rfloor$.

        if $|\, \pi_k((\mathbf{c}_u - \mathbf{c}_v)\, \mathbf{B})\, | > R$ then
            return the empty set $\varnothing$                  ▷ Prune the search tree.
        end if

        Let $\mathcal{S} =$ EnumerateInner($\mathbf{c}_u$, $k-1$).          ▷ Descend into branch.

        loop
            Let $u_k = u_k + 1$.
            if $|\, \pi_k((\mathbf{c}_u - \mathbf{c}_v)\, \mathbf{B})\, | > R$ then
                break the loop                   ▷ No need to consider greater $u_k$.
            end if
            Let $\mathcal{S} = \mathcal{S} \cup$ EnumerateInner($\mathbf{c}_u$, $k-1$). ▷ Descend into branch.
        end loop

        Let $u_k = \hat{u}_k$ as defined above.

        loop
            Let $u_k = u_k - 1$.
            if $|\, \pi_k((\mathbf{c}_u - \mathbf{c}_v)\, \mathbf{B})\, | > R$ then
                break the loop                   ▷ No need to consider lesser $u_k$.
            end if
            Let $\mathcal{S} = \mathcal{S} \cup$ EnumerateInner($\mathbf{c}_u$, $k-1$). ▷ Descend into branch.
        end loop

        return $\mathcal{S}$.
    end procedure

    return EnumerateInner($\mathbf{0}$, $D$).
end procedure
```

Algorithm 8 accepts as input a reduced basis $\mathbf{B}$ for the lattice L and a vector $\mathbf{v} \in L$. It returns the set of all vectors within distance R of $\mathbf{v}$. It is assumed that each recursive call to EnumerateInner generates a new isolated context for $\mathbf{c}_u = (u_1, \ldots, u_D)$ and k. Furthermore, it is assumed that assigning u_k changes the k^{th} component of $\mathbf{c}_u$ within the current context.

In practice, when searching for a discrete logarithm d, it is not necessary to construct the set of all vectors within distance R of $\mathbf{v}$. Rather, it suffices to test for each vector whether its last component is d.