A Quantum "Magic Box" for the Discrete Logarithm Problem

Burton S. Kaliski Jr.*

Version 1.3 — August 1, 2017

Abstract

In their foundational paper on pseudorandom bit generation, Blum and Micali showed that the discrete logarithm problem could be solved efficiently given a "magic box" oracle that computes the most significant bit of the discrete logarithm with a slight advantage over guessing. This magic box can be realized on a quantum computer with a new, simplified variant of Shor's algorithm. The resulting combination of Blum and Micali's reduction and this new quantum magic box offers an intriguing hybrid approach to solving the discrete logarithm problem with a quantum computer. Because the only requirement on the quantum portion of the algorithm is that it provide an approximate estimate of a single bit of the discrete logarithm, the new algorithm may be easier to implement, more resilient to errors, and more amenable to optimization than previous approaches. Further analysis is needed to quantify the extent of these benefits in practice. The result applies to the discrete logarithm problem over both finite fields and elliptic curves.

1 Introduction

The steady stream of advances in quantum computing is drawing increased attention to one of cryptography's fundamental long-term questions: When will a quantum computer be built that can break full-scale cryptosystems based on discrete logarithms and integer factoring?

Shor showed in his breakthrough paper [Sho99] that both problems can be solved efficiently on a quantum computer. However, the potential for a break of today's public-key cryptosystems is so far only a concern in theory. Although small-scale quantum computers have been built, scaling them to cryptographic parameters of the size currently employed in practice still requires major advances. Nevertheless, "enough progress is being made," according to a January 2016 NSA document [NSA16], to motivate the near-term adoption

*bkaliski@alum.mit.edu. The views expressed are my own and do not necessarily reflect those of my employer.

of new public-key algorithms resistant to quantum computing, particularly for systems that will be used "for many decades in the future." NIST has recently embarked on a "post-quantum" project that will lead to the selection of one or more such algorithms [NIS16].

A potential practical challenge of Shor's algorithm and its successors over the past two decades may be that they are asking a quantum computer to do too much: to solve essentially the entire discrete logarithm in a single run of a quantum algorithm. It may be better to ask instead about the *minimum* that a quantum computer needs to do to make the discrete logarithm problem solvable. As it turns out, the question has a well established answer in Blum and Micali's paper on pseudorandom bit generation [BM84]. Blum and Micali showed that a classical algorithm could solve the discrete logarithm problem given an oracle that estimates the most significant bit or "half-bit" of the discrete logarithm with a slight advantage over guessing. Such a "magic box," as they called it, is all that is needed. A minimum goal for a quantum algorithm for the discrete logarithm thus follows: estimate the half-bit.

Previous research has developed a number of improvements to Shor's algorithm as well as algorithms for problems such as quantum phase estimation, including various techniques based on processing partial information about the quantum phase [Kit95] [CEMM98] [ME99] [KSV02] [AC10] [PJC+14] [SHF14]. Very recently, Roetteler *et al.* [RNSL17] further built on these approaches to provide concrete cost estimates for solving the elliptic curve discrete logarithm problem on a quantum computer, again estimating the full phase and solving for the full discrete logarithm within a single quantum circuit. However, direct quantum estimation of the half-bit of the discrete logarithm does not appear to have been considered previously. Guedes *et al.* [GdAL+10][GdAL13] have studied a different (and apparently harder) problem related to Blum and Micali's pseudorandom generator: how to determine the underlying state of the generator from its output bits. Although Guedes *et al.*'s algorithm could be used in principle to validate a guess of a new output bit and thereby estimate the half-bit, its complexity is exponential — it is again attempting to do too much (at least for present purposes).

The rest of this paper is organized as follows. Section 2 provides definitions and notation for the discrete logarithm problem and half-bit approximation. Section 3 summarizes a general approach to solving the discrete logarithm problem on a quantum computer, based on the improvement to Shor's algorithm developed by Mosca and Ekert [ME99]. Section 4 provides the main contribution of the paper: a quantum algorithm for approximating the half-bit. The success probability of this new quantum magic box is analyzed in Section 5. Section 6 proposes several improvements, and Section 7 concludes the paper. Mosca and Ekert's improvement to Shor's algorithm is described in more detail in Appendix A.

2 Discrete logarithm problem and half-bit approximation

Let G be a group, and let a be an element of G of order r. Let b be a power of a. Then the *discrete logarithm problem* over G is to determine, given a and b, the unique solution m to the equation

$$b = a^m,\ 0 \leq m < r\,.$$

(Here and elsewhere, the group operation is written multiplicatively.) Let $\ell = \lfloor \log_2 r \rfloor + 1$ be the length in bits of r.

The *half-bit* of the discrete logarithm, denoted $\mathrm{HB}_a(b)$, is defined as follows:

$$\mathrm{HB}_a(b) = \begin{cases} 0 & \text{if } 0 \leq m < r/2, \\ 1 & \text{if } r/2 \leq m < r. \end{cases}$$

Blum and Micali showed that if an oracle could approximate $\mathrm{HB}_a(b)$ correctly with probability $1/2 + \epsilon$ on a random target b, then a reduction algorithm could solve the discrete logarithm problem with polynomially many (in ℓ and ϵ^{-1}) queries to the oracle on a variety of related values. Their result assumed that r is even, and was described for the case where G is the multiplicative group of a finite field. The reduction was extended to the case that r is odd by Oded Goldreich, and applied to elliptic curve groups in subsequent work [Kal86][Kal88].

In the following, the order r will be assumed to be an odd prime, as is typical in today's cryptography. This also simplifies the algorithm description (e.g., if r is prime then almost all values are invertible modulo r, etc., so "special cases" can generally be ignored). The algorithm descriptions could be expanded to cover non-odd-prime values r, with some technical adjustments. However, the non-odd-prime case may not be needed in practice. In particular, given an efficient algorithm for solving the discrete logarithm problem in a cyclic group when the base a has an arbitrary odd prime order, it's possible to solve the discrete logarithm problem with respect to any base in the group by decomposing into a series of discrete logarithm problems on prime-order bases, and then recomposing the results. (Discrete logarithms are trivial to compute when the order of the base a is a power of 2.)

Given the focus on the case where r is odd, a brief description of Goldreich's reduction will help ground the results in this paper. Goldreich's reduction, like Blum and Micali's, assumes that the oracle is correct with probability $1/2 + \epsilon$ on average. This doesn't mean that the oracle is necessarily correct on every *specific* input with that probability; if it were, then it would sufficient just to query the oracle enough times for the advantage to show up as a visible preference toward 0 or 1 outputs. Rather, it means that the oracle is correct overall with this advantage. Goldreich's reduction, similar to Blum and Micali's, *amplifies*

the advantage by making queries on related targets of the form $a^s b$ for random, known values of s. Assuming the discrete logarithm of b is very close to either $c/2$ or $c/2 + r/2$ for some known value c, the response to the query will be correlated with the half-bit of $a^{s+c/2}$ in the first case, or with its complement, in the second. ("Very close" means within a range that's on the order of ϵ itself; "half-bit" of $a^{s+c/2}$ should be read as distinguishing whether $0 \leq s + c/2 < r/2$, or $r/2 \leq s + c/2 < r$, all values being evaluated modulo r.) The two cases can then be distinguished with probability $1 - \delta$ with $O(\epsilon^{-1}\delta^{-2})$ queries. The slight advantage is thus amplified to a near-complete advantage.

Building on this advantage, the reduction solves for the discrete logarithm one bit at a time. In particular, starting with the assumption that the discrete logarithm of b^{2^ℓ} is very close to some known value c, it follows that the discrete logarithm of $b^{2^\ell - 1}$ is very close to either $c/2$ or $c/2 + r/2$. These two cases can be distinguished by the amplification just described (centering around $b^{2^\ell - 1}$ rather than b). The reduction continues recursively, the closeness improving by a factor of 2 each time, until the discrete logarithm of $b^{2^0} = b$ is uniquely determined. However, all this assumes the initial guess of c is close enough. The process is thus repeated $O(\epsilon^{-1})$ times on different initial guesses of c until the discrete logarithm is found. If $\delta = \ell^{-1}/2$, then the probability that all ℓ bits are solved correctly is at least $1/2$. Overall, the reduction therefore requires $O(\epsilon^{-2}\ell^2)$ queries.

3 Discrete logarithms on a quantum computer

In typical cryptographic applications, the group G is either a multiplicative subgroup of a finite field or a subgroup of an elliptic curve group over a finite field. In both cases, the currently recommended minimum size ℓ of the order r for long-term security (beyond the year 2030) is 256 bits [Bar16]. (In the first case, the minimum size of the finite field itself, i.e., the modulus, is 2048 bits.) The fastest known classical algorithms for solving the discrete logarithm problem in these cases take roughly 2^{128} operations, and the complexity grows exponentially in ℓ (as well as subexpoentially in the size of the modulus in the finite field case). In contrast, as Shor showed, a quantum algorithm can solve the discrete logarithm problem in time *polynomial* in ℓ. Thus, even if discrete logarithm cryptosystems remain secure against classical computing through the year 2030 and beyond, they may well be breakable in that timeframe if a sufficiently large-scale quantum computer can be built. The exact complexity and scale required of such a computer will depend on many implementation factors, motivating detailed analysis and optimizations.

Shor's algorithm combines group exponentiation operations and the Quantum Fourier Transform (QFT) to estimate, in effect, the *quantum phase shift* associated with multiplying a specially constructed quantum superposition by the group element b. Given the quantum phase shift angle, it is straightforward to recover the discrete logarithm. Mosca

and Ekert provide an excellent description of an optimized algorithm this performs this process [ME99]. Their algorithm has two stages. In the first stage, the algorithm places a quantum register in approximation of the superposition $|\Psi_k\rangle$ defined as

$$|\Psi_k\rangle = \frac{1}{\sqrt{r}} \sum_{t=0}^{r-1} \exp(-2\pi i k t/r) \, |a^t\rangle \, ,$$

where k is a known value between 0 and $r-1$. In the second, the algorithm estimates the quantum phase shift due to multiplying $|\Psi_k\rangle$ by the target b. The discrete logarithm is then solved by a straightforward classical computation on k and the quantum phase shift angle θ. Appendix A gives a more detailed description.

The basic observation on which Mosca and Ekert's algorithm is based is that multiplying the superposition $|\Psi_k\rangle$ by b — in their notation, applying the operator U_b — is equivalent to a quantum phase shift by the exponential $\exp(2\pi i\theta)$ where $\theta = km/r$. (Here, for simplicity, quantum phase shift angles are normalized to the range $[0, 1)$.) The superposition $|\Psi_k\rangle$ is thus an *eigenstate* of the operator U_b; the exponential $\exp(2\pi i\theta)$ is the corresponding *eigenvalue*, or "phase kickback." Given an accurate estimate of the angle θ, it is easy to determine $(km \mod r)$, and thereby solve for m.

Shor's algorithm and subsequent improvements face a common challenge: they need an accurate ℓ-bit estimate of the angle θ in order to solve for m. Interestingly, however, the various approaches to quantum phase estimation all generally build up the estimate one bit at a time. Indeed, even the QFT itself effectively consists of ℓ single-bit approximations, as Mosca and Ekert elegantly demonstrate with their "flying qubits" optimization [ME99]. Rather than the full QFT estimating all ℓ qubits in parallel, they estimate one bit at a time, controlling the internal operations of the QFT "semi-classically" based on previous outputs.

Kitaev introduced a novel non-QFT approach where multiple estimates for individual bits are assembled via a classical algorithm into an overall, accurate ℓ-bit estimate [Kit95]. This approach is further developed by Kitaev, Shen and Vyalyi [KSV02]; Ahmadi and Chiang [AC10]; and Svore, Hastings and Freedman [SHF14]. Patil *et al.* give a helpful summary and comparison of these and other related approaches [PJC+14]. However, despite the single-bit approximations, none of the methods appears to meet the requirement for Blum and Micali's magic box, because the partial information they produce is about the value $(km \mod r)$, not m itself. In particular, to get the half-bit of m seems to require nearly a full ℓ-bit approximation, followed by a multiplication by $(k^{-1} \mod r)$ to recover m. The previous algorithms thus only appear to contribute to the solution of the discrete logarithm problem if they are run to completion — all ℓ bits — and if that completion is accurate enough.

The idea of approximating a single bit of the quantum phase shift angle of an operator

such as U_b nevertheless suggests a way forward to estimating the half-bit, based on the phase of a different operator.

4 A quantum magic box for half-bit approximation

Suppose that instead of the target b, the input to the second stage were the group element b' defined as

$$b' = b^{kInv} ,$$

where the exponent $kInv$ is defined as

$$kInv = k^{-1} \mod r .$$

The quantum phase shift corresponding to multiplying the superposition $|\Psi_k\rangle$ by b' would then be $\exp(2\pi i\theta')$ where $\theta' = km'/r$ and m' is the discrete logarithm of b', i.e.,

$$m' = kInv \cdot m \mod r .$$

It follows that the quantum phase shift angle θ' for the operator $U_{b'}$ on eigenstate $|\Psi_k\rangle$ satisfies

$$\theta' \equiv km'/r \equiv k \cdot kInv \cdot m/r \equiv m/r \pmod 1 ,$$

or equivalently that $\exp(2\pi i\theta') = \exp(2\pi im/r)$. The dependence on k has been removed, and partial information about the angle θ' now directly relates to m. Thus, the half-bit of m can be estimated with a single-bit approximation of θ' with a modified version of Mosca and Ekert's second stage in Appendix A.2.

The modified second stage involves the operator U_{b^x} but has three registers rather than two. The first register has a fixed value, $kInv$, so it can be realized with classical bits; the value is denoted in bra-ket notation as $|kInv\rangle$ for convenience. The second register is again assumed to be in the superposition $|\Psi_k\rangle$; the analysis in Section 5 will show that the overall algorithm is still effective in an actual run where the second register is in an approximate superposition $|\tilde{\Psi}_y\rangle$. The third register is an additional control bit to the operator.

The steps of the modified second stage are as follows.

1. Compute $kInv = k^{-1} \mod r$. This step can be done classically.
2. Initialize the first register to the ℓ-bit state $|kInv\rangle$. The second register continues in the superposition $|\Psi_k\rangle$. Initialize the third register to the one-qubit state $|0\rangle$:

$$|kInv\rangle\,|\Psi_k\rangle\,|0\rangle \ .$$

3. Apply a one-qubit Hadamard transform to the third register. This places the third register in the superposition of two one-bit states, so the overall state becomes

$$|kInv\rangle\,|\Psi_k\rangle\left(\frac{1}{\sqrt{2}}\,|0\rangle+\frac{1}{\sqrt{2}}\,|1\rangle\right).$$

4. If the third register is $|1\rangle$, apply the operator U_{b^x} to the second register where x is the value of the first register. Because $x = kInv$, U_{b^x} has the same effect as $U_{b'}$. Then apply a quantum phase shift of $\exp(-\pi i/2) = -i$ to the third register (i.e., rotate by this amount when the third register is $|1\rangle$). This places the three registers in the entangled superposition

$$\frac{1}{\sqrt{2}}\,|kInv\rangle\,|\Psi_k\rangle\,|0\rangle-\frac{i}{\sqrt{2}}\,|kInv\,\rangle\,U_{b'}(|\Psi_k\rangle)\,|1\rangle\,.$$

   As shown above, the effect of the operation $U_{b'}$ on the eigenstate $|\Psi_k\rangle$ is the same as a quantum phase shift of $\exp(2\pi im/r)$. The resulting superposition is therefore equivalent to

$$\frac{1}{\sqrt{2}}\,|kInv\rangle\,|\Psi_k\rangle\,|0\rangle-\frac{i}{\sqrt{2}}\,\exp(2\pi im/r)\,|kInv\rangle\,|\Psi_k\rangle|1\rangle\,.$$

5. Apply a one-bit Hadamard transform to the third register again. This produces the superposition

$$\frac{1}{2}\Big(1-i\exp(2\pi im/r)\Big)\,|kInv\rangle\,|\Psi_k\rangle\,|0\rangle+\frac{1}{2}\Big(1+i\exp(2\pi im/r)\Big)\,|kInv\rangle\,|\Psi_k\rangle\,|1\rangle\,.$$

6. Measure the third register, obtaining either $|0\rangle$ or $|1\rangle$.

As in Mosca and Ekert's second stage in Appendix A.2, the operator U_{b^x} can be implemented with a series of controlled multiplications by successive squares of b, as shown in Figure 1. Here, each multiplication is doubly controlled by the corresponding bit of the first register and the common control bit of the third. Because the first register has only a single value, its control of the multiplier can also be realized semi-classically; the bits of $kInv$ simply determine which multipliers are involved in implementing U_{b^x}. Thus, the number of qubits for the modified second stage in this basic form is the same as in the "flying qubits" optimization for Mosca and Ekert's second stage, but there's only one measurement.

Assuming that the second register is initially in the eigenstate $|\Psi_k\rangle$ at the start of the modified second stage, the probability distribution of the third register is

$$\Pr[|0\rangle]=\left\|\frac{1}{2}(1-i\exp(2\pi im/r))\right\|^2=\frac{1}{2}+\frac{1}{2}\sin(2\pi m/r)\,;$$

$$\Pr[|1\rangle]=\left\|\frac{1}{2}(1+i\exp(2\pi im/r))\right\|^2=\frac{1}{2}-\frac{1}{2}\sin(2\pi m/r)\,.$$

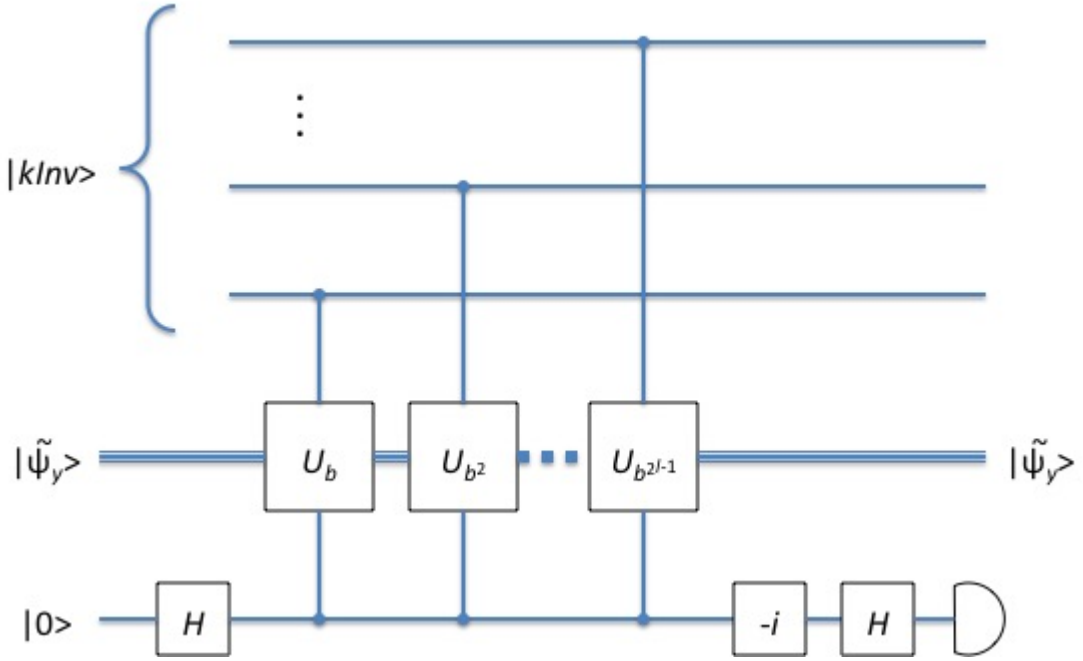

Figure 1: Modified second stage — the quantum magic box. H denotes a one-qubit Hadamard transform, $U_b, U_{b^2}, \ldots$ denote controlled multiplications by successive squares of b, and $-i$ denotes a phase shift of $\exp(-\pi i/2)$. The third register when measured provides an estimate of the half-bit of the discrete logarithm of b.

The success probability of the modified second stage in estimating the half-bit of the discrete logarithm m is thus $1/2 + \epsilon_m$, where

$$\epsilon_m = (1/2)\left|\sin(2\pi m/r)\right| .$$

The average advantage ϵ of the algorithm on a random query b can be approximated as

$$\epsilon = \frac{1}{r}\sum_{m=0}^{r-1} \epsilon_m \approx \frac{1}{r}\int_0^r \frac{1}{2}\left|\sin(2\pi m/r)\right| dm = \frac{1}{r}\int_0^{r/2} \sin(2\pi m/r)\, dm = \frac{1}{\pi} .$$

The resulting two-stage algorithm is thus correct on average with probability roughly $1/2 + 1/\pi \approx 82\%$ for a random b, a very good approximation of the half-bit. The algorithm has become Blum and Micali's magic box, at least in the ideal case. Additional analysis is needed to confirm that the algorithm still has a sufficient advantage in an actual run where the ideal eigenstate is approximated by $|\tilde{\Psi}_y\rangle$.

Remark. In contrast to Shor's algorithm, the output of the quantum magic box can't be checked immediately (unless the answer is known in advance). The correctness is confirmed instead in the context of the reduction from the discrete logarithm problem, after accumulation and analysis of the output for many queries.

5 Success probability of an actual run

Let $|\xi\rangle$ be an arbitrary superposition defined as

$$|\xi\rangle = \frac{1}{\sqrt{A}} \sum_{t=0}^{r-1} \alpha_t \, |a^t\rangle \,,$$

where the value A is defined as

$$A = \sum_{t=0}^{r-1} \|\alpha_t\|^2,$$

so that $\||\xi\rangle\|^2 = 1$.

If the second register in the modified second stage is initialized to the state $|\xi\rangle$, then the state after the controlled multiplication by b' and the quantum phase shift of $\exp(-\pi i/2)$ will be

$$\frac{1}{\sqrt{2}} \, |kInv\rangle \, |\xi\rangle \, |0\rangle - \frac{i}{\sqrt{2}} |kInv\rangle \, U_{b'}(|\xi\rangle) \, |1\rangle \,.$$

The effect of applying $U_{b'}$ to $|\xi\rangle$ is to shift indices by m':

$$U_{b'}(|\xi\rangle) = U_{b'}\Big(\sum_{t=0}^{r-1} \alpha_t \, |a^t\rangle\Big) = \sum_{t=0}^{r-1} \alpha_{t-m'} \, |a^t\rangle \,.$$

(Here and in the following, subscripts of α are evaluated modulo r.) The state at the end of the modified second stage, following the one-bit Hadamard transform, will thus be

$$\frac{1}{2\sqrt{A}} \sum_{t=0}^{r-1} (\alpha_t - i\alpha_{t-m'}) \, |kInv\rangle \, |a^t\rangle \, |0\rangle + \frac{1}{2\sqrt{A}} \sum_{t=0}^{r-1} (\alpha_t + i\alpha_{t-m'}) \, |kInv\rangle \, |a^t\rangle \, |1\rangle \,.$$

The probability that the third register will be measured in the state $|0\rangle$ is therefore

$$\begin{aligned}
\Pr[|0\rangle] &= \frac{1}{4A} \sum_{t=0}^{r-1} \|\alpha_t - i\alpha_{t-m'}\|^2 \\
&= \frac{1}{4A} \sum_{t=0}^{r-1} (\alpha_t - i\alpha_{t-m'}) \overline{(\alpha_t - i\alpha_{t-m'})} \\
&= \frac{1}{4A} \Big(\sum_{t=0}^{r-1} \alpha_t \overline{\alpha_t} + i \sum_{t=0}^{r-1} \alpha_t \overline{\alpha_{t-m'}} - i \sum_{t=0}^{r-1} \alpha_{t-m'} \overline{\alpha_t} + \sum_{t=0}^{r-1} \alpha_{t-m'} \overline{\alpha_{t-m'}} \Big) \,.
\end{aligned}$$

The first and fourth terms each sum to A, while the middle terms are each based on the *autocorrelation* of the sequence α_t at lag m', defined as

$$R_{\alpha\alpha}(m') = \sum_{t=0}^{r-1} \alpha_t \overline{\alpha_{t-m'}} \,. \tag{1}$$

(Note that $A = R_{\alpha\alpha}(0)$.) The probability thus simplifies to

$$\Pr[|0\rangle] = \frac{1}{2} + \frac{i}{4A}\Big(R_{\alpha\alpha}(m') - \overline{R_{\alpha\alpha}(m')}\Big) \,.$$

Now consider an actual run of the algorithm where the second register is in the superposition $|\tilde{\Psi}_y\rangle$ measured in the first stage (see Appendix A.1) for a particular frequency y. As expressed in Section 4, the superposition $|\tilde{\Psi}_y\rangle$ is the sum of 2^ℓ terms, where, because $2^\ell > r$, some of the underlying states $|a^x\rangle$ are included more than once. Rewritten as a sum of r terms where each underlying state is included once, the superposition can be expressed as

$$|\tilde{\Psi}_y\rangle = \frac{1}{\sqrt{\tilde{A}_y}} \sum_{t=0}^{r-1} \alpha_t \, |a^t\rangle \,,$$

where the coefficient α_t is the sum of either one or two underlying exponentials:

$$\alpha_t = \begin{cases} \beta_t + \beta_{t+r} & \text{if } 0 \le t \le 2^\ell - r - 1, \\ \beta_t & \text{if } 2^\ell - r \le t \le r - 1. \end{cases}$$

and where $\beta_x = \exp(-2\pi i y x / 2^\ell)$. The autocorrelation may be rewritten similarly as a sum of cross-products of underlying exponentials $\beta_u \overline{\beta_v}$:

$$R_{\alpha\alpha}(m') = \sum_{\substack{0 \le u,v \le 2^\ell - 1 \\ (u-v) \equiv m' \pmod r}} \beta_u \overline{\beta_v} \,. \tag{2}$$

Assuming as previously that r is an ℓ-bit number, i.e., $2^{\ell-1} \le r < 2^\ell - 1$, there will be either one or two values of v for which $(u - v) \equiv m' \pmod r$ for each value of u, and thus one or two cross-products $\beta_u \overline{\beta_v}$ for each u.

The advantage of the algorithm on a query $b = a^m$ depends on the ratio

$$\frac{i(R_{\alpha\alpha}(m') - \overline{R_{\alpha\alpha}(m')})}{4R_{\alpha\alpha}(0)} \,.$$

In the ideal case where the second register is in the eigenstate $|\Psi_k\rangle$, the first autocorrelation sum in the numerator will have r cross-product terms, each with value $\alpha_t \overline{\alpha_{t-m'}} =$

$\exp(2\pi im/r)$. The second autocorrelation sum will involve r terms each with the conjugate value $\exp(-2\pi im/r)$. The numerator will thus equal $i\cdot(r\exp(2\pi im/r) - r\exp(-2\pi im/r)) = 2r\sin(2\pi m/r)$. The denominator will equal $4r$. (This gives another derivation of the the ideal advantage $(1/2)|\sin(2\pi m/r)|$ in the previous section.) In an actual run, however, with the calculation based on cross-products $\beta_u\overline{\beta_v}$, both the number of cross-products in each sum and their values will vary from the ideal. The actual average advantage may therefore be less than the ideal $1/\pi$. The following subsections analyze by how much.

5.1 Variation in number of cross-products

Observe first that the autocorrelation $R_{\alpha\alpha}(0)$ following Equation (2) is the sum of $2^{\ell+1} + 2^\ell - 2r$ cross-products $\beta_u\overline{\beta_v}$: two each for the first and last $2^\ell - r$ values of u, and one each of the remaining $2r - 2^\ell$ values. It follows that $R_{\alpha\alpha}(0) \le 2^{\ell+1} + 2^\ell - 2r$. (Indeed, the exact value can be shown to be $2^\ell + 2(2^\ell - r)\cos(2\pi yr/2^\ell)$.)

An autocorrelation at lag $m' > 0$ will have fewer cross-products, because the "peaks" where both α_t and $\alpha_{t-m'}$ contribute two underlying exponentials will not be as well aligned. The minimum total number of cross-products for any m' depends on r. If $2^{\ell-1} < r < 2^{\ell+1}/3$, then the minimum is $2^{\ell+2} - 4r$. If $2^{\ell+1}/3 < r < 2^\ell$, then the minimum is $2^{l+1} - r$. The minimum ratio between the lower bound on the number of terms in the autocorrelation sum for $m' > 0$ and in the upper bound at $m = 0$ occurs when $r \approx 2^{\ell+1}/3$, which produces a minimum ratio of $4/5$. Thus, the advantage will be at least $4/5$ of the ideal advantage, after reducing by errors within the actual cross-products described next.

5.2 Errors within cross-products

Define $\zeta_y = yr/2^\ell - k$. The phase error between the exponential β_x and the ideal value $\exp(-2\pi ikx/r)$ is $-2\pi\zeta_y x/r$. The phase error in each exponential is thus bounded in absolute value by $(2\pi 2^\ell/r)|\zeta_y|$. Because the phase errors are all in the same direction, the phase errors in the cross-products $\beta_u\overline{\beta_v}$ are thus also bounded in absolute value by $\Delta_{\text{freq}} = (2\pi 2^\ell/r)|\zeta_y|$.

If the quantum algorithm is implemented with an Approximate FFT (AFFT) of degree d instead of the QFT, then, following [Cop02], the underlying exponentials will each have an additional phase error that is bounded in absolute value by $\Delta_{\text{AFFT}} = (2\pi\ell)/2^d$. Because the phase errors are again all in the same direction, the same bound also applies to the phase error per cross-product.

5.3 Overall impact

A phase error in a cross-product translates to the same phase error in the argument to the sine, and thus in the worst case the same error in the output of the sine. Considering the 4/5 scaling factor and the error bounds above, this means that the advantage on a random query b in an actual run is at least

$$\frac{4}{5} \cdot \Big(\frac{1}{\pi} - \frac{1}{2} \Big(\frac{2\pi 2^\ell}{r} |\zeta_y| + \frac{2\pi\ell}{2^d} \Big) \Big) .$$

This advantage can still be significant. For example, if $|\zeta_y| \leq r/(8\pi 2^\ell)$ and $d \geq \log_2(\pi\ell)+3$, then the advantage will be at least

$$\frac{4}{5} \cdot \Big(\frac{1}{\pi} - \frac{1}{2} \Big(\frac{1}{4} + \frac{1}{4} \Big) \Big) \approx \frac{1}{20} . \tag{3}$$

The upper bound on $|\zeta_y|$ can be achieved by repeating the first stage until a suitable value of y is produced and only then proceeding to the second stage. Such values of y occur frequently. To see this, consider that the approximation error ζ_y is a multiple of $1/2^\ell$ in the range $[-1/2, 1/2)$. Because r is odd, there is a one-to-one correspondence between values of ζ_y in this range and values of y between 0 and $2^\ell - 1$. Moreover, values of y where ζ_y is closer to 0 are more likely to occur than others, because the relative probability of the superposition $|\tilde{\Psi}_y\rangle$ is higher in such cases (note the term $\cos(2\pi y r/2^\ell) = \cos(2\pi\zeta_y)$ in the formula for $\tilde{A}_y$ above).

As a result, the probability that the first stage will produce a value of y where $|\zeta_y| \leq r/(8\pi 2^\ell)$ is at least $r/(4\pi 2^\ell)$. Because $r \geq 2^{l-1}$, the probability is at least $1/(8\pi)$, so at most 8π runs of the first stage would be required on average.

As a concrete example, if $\ell = 256$, the minimum AFFT degree to meet the bound $d \geq \log_2(\pi\ell) + 3$ would be $d = 13$.

6 Potential improvements

The description just given shows the feasibility of the quantum magic box with a worst-case analysis and without optimization. Various improvements in both the features of the algorithm and its analysis may well yield an even better algorithm. Some of the possible approaches are detailed here, and are offered for further analysis. The reader may well be able to find others.

6.1 Additional filtering

One potential improvement to the quantum magic box would be to run the first stage even more times than proposed above, until the resulting value $|\zeta_y|$ is arbitrarily small. The number of runs is inversely proportional to the desired upper limit on the value. For example, to get $|\zeta_y|$ below $r/(64\pi 2^\ell)$, so that the impact on the advantage due to frequency approximation is bounded by $1/64$ rather than $1/8$, at most 64π runs of the first stage would be required on average. The improved advantage could reduce the overall number of queries to the quantum magic box, counterbalancing the increase in the number of runs of the first stage. Alternatively, the improvement to $|\zeta_y|$ could make it possible to accommodate a smaller AFFT degree d without significantly affecting the overall advantage.

6.2 Range restriction

Blum and Micali developed their discrete logarithm reduction to prove that their pseudorandom bit generator was indistinguishable (in polynomial time) from a truly random generator. They therefore assumed the weakest possible scenario that would violate indistinguishability: that an adversary could predict bits of the generator with a non-negligible advantage over guessing, on average. For this reason, Blum and Micali's analysis focuses on the average advantage on a random input.

The quantum magic box presented here does much better than average on *specific* inputs, particularly those whose discrete logarithms are close to $r/4$ and $3r/4$ (where the sine value is close to ± 1). Thus, the efficiency of the reduction can be improved if queries are restricted to this *high-advantage* range. (The restriction can be achieved with a minor adjustment to the reduction.)

Recall that the ideal advantage of the algorithm is $\epsilon_m = (1/2)\big|\sin(2\pi m/r)\big|$. If m is near $r/4$ or $3r/4$, the advantage will be nearly 50%, versus the average of $1/\pi \approx 32\%$ for a random m — an improvement of 18%. This provides substantial "head room" for additional approximation errors. It also provides further opportunity for optimizations in the algorithm, by trading off between the width of the range and the average advantage within the range, both of which affect the number of queries.

6.3 Generic multiplication

The modified second stage in the basic quantum magic box (see Figure 1), similar to the second stage of Mosca and Ekert's algorithm, involves ℓ controlled multipliers. This is convenient because the circuit can be optimized specifically for b, but it produces a circuit of depth $O(\ell)$ (where depth is calculated in terms of the number of multipliers). In Mosca

and Ekert's second stage, this is reasonable, given that the stage also includes a QFT and that each multiplier is applied independently to compute the superposition

$$\frac{1}{2^{\ell/2}} \sum_{x=0}^{2^\ell - 1} |x\rangle \, U_{b^x}(|\Psi_k\rangle) \, .$$

All possible ℓ-qubit values x are included in the sum. In the modified second stage, in contrast, only two ℓ-qubit values are included in the sum, $x = 0$ and $x = kInv$. The circuit depth could thus be reduced to $O(1)$ by including only a controlled multiplier for $U_{b'}$. However, constructing the second stage this way is problematic, because the value of b' is not known until *after* the first stage is run and the values of y and thus k and $kInv$ are determined. The second stage would thus have to be constructed while the second register remains in superposition.

A possible alternative would be to include a *generic* multiplier and to make the value b', which can be computed classically, an input to the second stage. The generic multiplier would perform the invertible map $(b', |\Psi_k\rangle) \mapsto (b', U_{b'}(|\Psi_k\rangle))$. The multiplier may also need the inverse $(b')^{-1}$; this could be computed classically as well and also input to the second stage, or it could be computed within the second stage. For example, in the case that the group G is a multiplicative subgroup modulo a prime, the inverse could be computed via Proos and Zalka's quantum circuit for the extended Euclidean algorithm [PZ03]. If G is an elliptic curve group, inverse computation is generally trivial, e.g., just a change in the value of a sign, depending on the point representation.

A generic multiplier would be less efficient than a multiplier optimized for a fixed value such as b, b^2, etc. However, only one generic multiplier would be required, rather than $O(\ell)$ fixed multipliers. This design change would further reduce the burden on the quantum algorithm, and may also make it more practical to run the second stage on additional inputs, as described in the next subsection.

An intriguing extension of these ideas would be to include a generic *exponentiator* that raises the the second register to the exponent k at the end of the first stage. Such an exponentiator could be implemented with standard techniques in reversible computing, given both k and $kInv$. Given an input $|a^t\rangle$, the exponentiator would first compute $|a^{kt}\rangle$ along with intermediate values, then reverse the computation to erase the intermediate values, resulting in only $|a^t\rangle$, $|a^{kt}\rangle$, and zeros. The exponentiator would then compute $|(a^{kt})^{kInv}\rangle = |a^t\rangle$ and erase the original input value by combining it with this value. It would again reverse the computation to erase intermediate values, leaving only the desired result $|a^{kt}\rangle$. Might this provide a path toward computing the elusive eigenstate $|\Psi_1\rangle$? (Cleve *et al.* state that this state "is not at all trivial to fabricate" [CEMM98].) If the original input is the superposition $|\Psi_k\rangle$, then the output would apparently be $|\Psi_1\rangle$. Starting with $|\Psi_1\rangle$, the modified second stage could then be implemented with only a single multiplication by

b. Fibonacci exponentiation would be preferable to binary exponentiation for this purpose, because a reversible circuit for binary exponentiation would have to accumulate the $O(\ell)$ intermediate squarings in the binary "addition chain," whereas a reversible circuit for Fibonacci exponentiation would only need to maintain the most recent pair of adjacent values in the Fibonacci addition chain (see [Kal17] for further discussion).

6.4 Extending the second stage

Another potential improvement would be to extend the second stage to obtain additional estimates based on the same eigenstate. This could be done with additional sets of multipliers corresponding to the same value of b, in order to get additional estimates for the same target and thereby amplify any advantage on that target. It could also be done with additional sets of multipliers corresponding to different values of b, in order to process additional magic box queries within the same run of the second stage. Alternatively, the additional values of b' could be provided as additional inputs if the generic multiplier improvement is followed.

The additional estimates can be implemented with parallel or serial circuits, following similar architectures as for quantum phase estimation, e.g., in Svore *et al.*'s algorithm [SHF14]. The additional estimates could also produce information about additional bits of the discrete logarithm with appropriate choices of inputs. Note also that varying the first register (or multiplying the first register by an appropriate constant) can also facilitate estimates of different bits of m, e.g., if the first register were $(2kInv \mod r)$ then the measurement would provide an estimate of the "quarter-bit." The estimates for different bits could potentially be combined semi-classically with internal QFT phase shifts similar to Mosca and Ekert's "flying qubits" to improve the convergence toward m.

In contrast to previous algorithms, which also involve repeated estimates based on the same eigenstate, the repeated estimates here need only continue as long as they improve the performance of the algorithm; they don't need to solve for all ℓ bits. In particular, the motivation for extending the second stage is not because it's necessary to solve the discrete logarithm, but because it amortizes the cost of the first stage. This may be helpful if the first stage is run multiple times in order to find a good y.

The Blum-Micali / Goldreich reductions generate multiple queries of the form $a^s b^t$. This suggests another alternative: instead of constructing a different set of multipliers for each query, construct a single set and provide s and t (or, more precisely, $s \cdot kInv \mod r$ and $t \cdot kInv \mod r$) as inputs to the second stage. With this improvement, the second stage could be fully constructed just once given the base a and the overall target b. (The second stage would now also include multipliers by successive powers of a.)

Another reason to obtain an additional estimate in the second stage would be to test if the

algorithm is working correctly. One or more inputs whose discrete logarithms are known to be in the high-advantage range (see Section 6.2), could be tested. If the second stage produces an incorrect output on such a test value, then outputs from other estimates in the second stage could be discarded as a precaution.[1]

6.5 Multiple frequencies

Another design change that may improve the algorithm would be to skip the measurement of the frequency $|y\rangle$ in the first stage and instead perform the second stage on the superposition of frequencies, i.e., directly on the output of the QFT. Because all 2^ℓ possible frequencies would be in the superposition, the corresponding values of k and $kInv$ would also need to be computed within the quantum portion.[2] The latter can be computed via Proos and Zalka's extended Euclidean algorithm circuit [PZ03]; the case $k = 0$ can be handled by defining $kInv = 0$ to maintain reversibility, with negligible impact on the outcome.

A possible benefit of working with multiple frequencies simultaneously is that the advantage of the quantum magic box on a given query b would be influenced by the average effect of all frequencies involved, potentially bringing the advantage closer to the ideal. The filtering idea above could also be applied to focus on frequencies that have better approximations. In particular, the process of computing k from y can be expanded to produce a few bits of ζ_y. These bits could be maintained in a separate register, and measured when the first or second stage concludes. An estimate would only be accepted if the measurement corresponds to a sufficiently small ζ_y.

6.6 Analytical improvements

The most promising avenue for improvement is perhaps not in the algorithm itself but in its analysis. The discussion so far has focused on worst-case bounds. However, there is good reason to believe that the average case may be much better. In particular, although

[1] A related question for further work is modeling the evolution of the second register over the course of the multiple estimates. If the second register were in the eigenstate $|\Psi_k\rangle$, then it would in principle remain in the same eigenstate after each estimate. However, because the second register starts in the approximation $|\tilde{\Psi}_y\rangle$, it will evolve into in one of two new approximations after the estimate, depending on which way the third register is measured. Presumably, if the third register is measured with the correct half-bit, the second register will get closer to the eigenstate. If so, a series of correct answers on test values could potentially "prime" the second register so that it is even more accurate on a subsequent non-test value. However, any such theoretical improvement may well be overcome by the accumulation of operational errors over the course of multiple estimates.

[2] Computing k and $kInv$ within the quantum portion might also be convenient even when the frequency is measured, to reduce the number of classical interventions.

the scaling factor 4/5 will not move much (it depends on r), the error bounds $(2\pi 2^\ell)/r$ and $(2\pi\ell/2^d)|\zeta_y|$ are both extremes.

Recall that the bounds are derived from an upper limit on the *difference* between the phase errors of the two exponentials in the cross-product $\beta_u\overline{\beta_v}$. The average difference is likely to be much less. The following aspects would be worth exploring further:

- *Frequency approximation errors.* As discussed above, the absolute value of the phase error in a cross-product $\beta_u\overline{\beta_v}$ due to approximating the frequency k/r by $y/2^\ell$ ranges from 0 to an upper limit of $\Delta_{\text{freq}} = (2\pi 2^\ell/r)|\zeta_y|$. It is possible to show that the average absolute value is around $\Delta_{\text{freq}}/3$. Moreover, the errors in the various cross-products in the autocorrelation may partially cancel one another out. Thus, the worst-case value should be considered an overestimate.

- *AFFT approximation errors.* Following Coppersmith's analysis of the AFFT, the phase error in an individual exponential β_x due to the degree-d AFFT ranges from 0 to an upper limit of $\Delta_{\text{AFFT}} = (2\pi\ell)/2^d$. The phase error here is the result of potential errors in each of the ℓ bits of the QFT, each with individual impact at most $2\pi/2^d$. Making the perhaps equally extreme assumption that the phase errors associated with each of the ℓ bits are independently and uniformly distributed in the range $[0, \Delta_{\text{AFFT}}/\ell]$, the overall AFFT phase error would be approximately normally distributed with mean $\Delta_{\text{AFFT}}/2$. Its standard deviation would be $\sqrt{\ell}\cdot(\Delta_{\text{AFFT}}/\ell)/\sqrt{12} = \Delta_{\text{AFFT}}/\sqrt{12\ell}$.

  With the assumptions just stated, the expected phase error due to AFFT approximation in a cross-product $\beta_u\overline{\beta_v}$ would be 0 and its standard deviation would be $\sqrt{2}\cdot\Delta_{\text{AFFT}}/\sqrt{12\ell} = 2\pi\sqrt{\ell}/(\sqrt{6}\,2^d)$.

  Given such a distribution, a probabilistic approach may be taken to bound the error. For example, one could choose d such that the standard deviation $2\pi\sqrt{\ell}/(\sqrt{6}\,2^d)$ is no more than 7.5%. The probability that the absolute value of the error is greater than 15%, i.e., greater than two standard deviations, would then be less than 5%. Assuming that the impact on the output of the sine of the error has absolute value 2 outside when the error is outside the two standard deviation limit – a worst-case total reversal — the error due to the AFFT would be at most $15\% + 2\cdot 5\%$, which is within the 1/4 bound in Equation (3). If $\ell = 256$, this bound can be satisfied with degree $d = 9$, reducing the burden on the AFFT implementation by another four degrees. Of course, this all assumes a normal distribution, which may not be appropriate in the actual analysis. On the other hand, AFFT approximation errors in different cross-products in an autocorrelation may again partially cancel one another out. This again provides evidence that the worst case is an overestimate.

If the heuristic effects just discussed are reliable, then it should be possible to realize the quantum magic box with better parameters than initially proposed, and with better

performance. Further work is needed to confirm these bounds.

6.7 Improving existing algorithms

Another possible application of the "magic box" ideas motivated by the Blum-Micali / Goldreich reductions would be to improve the existing quantum algorithms for the discrete logarithm problem directly. For instance, rather than making repeated measurements of the phase shift associated with the same operator U_b, the algorithms could make multiple measurements of *related* operators $U_{a^s b}$ where the values of s are known and determined in advance. This is analogous to the amplification within the reductions, and means that the estimates don't need to be accurate for every specific b — just, once again, for the operators on average. The measurements would be correlated with s, with the assumption that b is in a sufficiently small range (and it may also be possible, heuristically, to use outputs from the same set of $U_{a^s b}$ operators repeatedly with different guesses of c, thus reducing the number of repetitions of the overall algorithm). The values of s could also be chosen from a small range similar to the suggestion above about range restriction, so that the amplification focuses on values that have better approximations (on average). The potential benefit of these improvements is similar to the quantum magic box in that they place fewer demands on the accuracy of the first stage, and of the quantum portion overall. However, the improved algorithms still must solve for the full discrete logarithm all in one run.

6.8 Distributed computation

Multiple quantum magic box implementations can collaborate on a solution to the discrete logarithm problem, each responsible for a subset of the queries. The distributed computation can provide a range of time-space tradeoffs, depending on how many of the $O(\epsilon^{-2}\ell^2)$ queries each implementation is given.

Recall that the Blum-Micali reduction involves $O(\epsilon^{-1})$ initial guesses c of the discrete logarithm of b^{2^ℓ}. A problem "orchestrator" can give different guesses c as starting points to different magic box implementations, along with the target a, and the implementations could work from the different guesses in parallel, generating a sequence of queries locally. On the other end of the spectrum, the orchestrator could generate all the queries itself and give individual queries to different implementations.

Implementations can also be given random queries with known discrete logarithms in order to test whether they are operating honestly. If a query is of the form $a^s b^t$ where s is random, then the query is effectively random, and thus indistinguishable from a random test query. For similar reasons, an implementation won't necessarily learn the target of the discrete

logarithm problem it's solving. By mixing test queries with actual queries, the problem orchestrator can verify an implementation's advantage, while also obtaining assurance that the answers to actual queries have a similar advantage over guessing.

7 Conclusion

Originally the core element of a security proof for a pseudorandom bit generation in a classical computing model, Blum and Micali's magic box turns out to be a prescient contribution to quantum algorithms for the discrete logarithm problem. The quantum magic box inspired by their reduction answers the question posed at the start of the paper: what is the minimum that a quantum algorithm needs to do in order to solve the discrete logarithm problem?

Although simpler than Shor's algorithm, the quantum magic box is still complex. It still needs the full first stage of the algorithm, so its overall scale is on the same order of magnitude. The second stage in its basic form involves a one-qubit Hadamard transform, ℓ doubly-controlled multiplications, a phase shift gate, and another one-qubit Hadamard transform. This is only slightly less than Mosca and Ekert's "flying qubits" method, which has, in addition to the ℓ controlled multiplications (singly controlled in this case) a total of 2ℓ one-bit Hadamard transforms all involving the same control qubit, plus $O(\ell^2)$ semi-classically controlled phase shift gates of various precisions. In effect, the savings are primarily in the number of measurements, and not the number of qubits. Although Kitaev's and other approaches take even more measurements, they can be optimized for various tradeoffs between circuit size and depth, depending on the parameter of interest.

However, there are at least three reasons to believe that the new algorithm may be easier to realize in practice.

First, it appears that the AFFT degree in the first stage can be significantly smaller, which means that the internal quantum phase shift gates don't need to be as precise. Whether a reduction in precision significantly reduces the cost of the quantum circuit, however, remains to be seen. Indeed, in Roetteler *et al.*'s detailed cost analysis, each quantum phase shift gate in the QFT is synthesized from a series of simpler, standard gates, while group operations are the overwhelming contributor to the circuit cost. Still, the lower the AFFT degree, the smaller the circuit, and perhaps with a small enough AFFT degree the quantum phase shift gates wouldn't even need to be synthesized, but could be realized directly in hardware.

Second, the algorithm has the potential to be more error-tolerant. As long as some non-negligible advantage over guessing is achieved on average, the algorithm can still contribute meaningful information to the solution of the discrete logarithm problem. The central

benefit of Blum and Micali's reduction is that it only requires that the magic box have an advantage on average given a random input — not that it have an advantage on any specific input. Although the various quantum phase estimation algorithms also amplify an advantage that may be slightly better than guessing, that advantage generally must be present for each of the measurements made. In contrast, for the quantum magic box, the estimates must be better than guessing just on average — a potentially more achievable result.

Third, as the discussion in the previous section illustrates, there are a large number of potential avenues for improvement given that the goal for the quantum portion has been lowered. These avenues add another dimension for optimization in addition to size and depth: the number of runs of the quantum portion.

It is possible that other quantum algorithms could be improved by similarly reducing the expectations on the quantum portion. It would be interesting to see if there's a way to do so for Shor's integer factoring algorithm, but the starting point would likely be different than proofs from the pseudorandom bit generation literature. Whereas the discrete logarithm problem has known reductions to single-bit approximation, the reductions for cryptosystems based on integer factoring relate inverting the RSA (or Rabin) function to approximating individual bits (see, e.g., [ACGS88]). Such reductions do not appear to lead to an improved quantum algorithm for integer factoring itself, which, at least following Shor's algorithm, would center on obtaining partial information about an unknown group order r. (Ekerå and Håstad [EH17] explore one possible line of simplification based on relating the unknown order of the RSA group to a short discrete logarithm problem.)

Many open questions remain about the line of research described in this paper, most importantly whether it would result in quantum algorithms that offer a meaningfully different answer to the fundamental question posed in the introduction. Would such improvements shorten the time until a quantum computer can practically break discrete logarithm cryptosystems? Or perhaps would they reduce the cost per solution? Such answers will need much more detailed analysis of quantum circuit designs and parameter choices, including tradeoffs between the amount of classical computation, i.e., number of steps in the discrete logarithm reductions, and the complexity and scale of the quantum portion. These are the kinds of questions cryptographers need to increase their attention to as the "post-quantum" era approaches, and no conclusion can be drawn without further analysis and possibly even implementation experience. Just as at the start of the "pre-quantum" era of cryptography, the answers are in the many details.

Acknowledgements

This research was done as a personal endeavor on many evenings and weekends, and I am grateful first to my wife, Michele, and to my son Steve and daughter Jessie for their encouragement to put the time into its development. Steve also contributed to some of the analysis. I also benefited from technical review by Michele Mosca, who provided helpful feedback on early versions. Finally, I am thankful for the many researchers who have devoted themselves to this field of technology, one that's so remarkably "entangled" into our universe. May we put the gifts they've uncovered to very good use.

References

[AC10] Hamed Ahmadi and Chen-Fu Chiang. Quantum phase estimation with arbitrary constant-precision phase shift operators. *arXiv preprint arXiv:1012.4727*, 2010.

[ACGS88] Werner Alexi, Benny Chor, Oded Goldreich, and Claus P Schnorr. RSA and Rabin functions: Certain parts are as hard as the whole. *SIAM Journal on Computing*, 17(2):194–209, 1988.

[Bar16] Elaine Barker. *NIST Special Publication 800-57: Recommendation for Key Management, Part 1: General*. NIST, Revision 4 edition, January 2016.

[BM84] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM Journal on Computing*, 13(4):850–864, 1984.

[CEMM98] Richard Cleve, Artur Ekert, Chiara Macchiavello, and Michele Mosca. Quantum algorithms revisited. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 454, number 1969, pages 339–354. The Royal Society, 1998.

[Cop02] Don Coppersmith. An approximate Fourier transform useful in quantum factoring. *arXiv preprint quant-ph/0201067*, 2002.

[EH17] Martin Ekerå and Johan Håstad. Quantum algorithms for computing short discrete logarithms and factoring rsa integers. In *International Workshop on Post-Quantum Cryptography*, pages 347–363. Springer, 2017.

[GdAL+10] Elloá B. Guedes, Francisco M. de Assis, Bernardo Lula Jr, Rua Aprígio Veloso, and Campina Grande-Paraíba-Brazil. Quantum permanent compromise attack to Blum-Micali pseudorandom generator. In *Proceedings of the International Telecommunications Symposium 2010*, 2010.

[GdAL13] Elloá B. Guedes, Francisco Marcos de Assis, and Bernardo Lula. Quantum attacks on pseudorandom generators. *Mathematical Structures in Computer Science*, 23(03):608–634, 2013.

[Kal86] Burton S. Kaliski Jr. A pseudo-random bit generator based on elliptic logarithms. In *Conference on the Theory and Application of Cryptographic Techniques*, pages 84–103. Springer, 1986.

[Kal88] Burton S. Kaliski Jr. *Elliptic curves and cryptography: A pseudorandom bit generator and other tools*. PhD thesis, Massachusetts Institute of Technology, 1988.

[Kal17] Burton S. Kaliski Jr. Targeted Fibonacci exponentiation. Manuscript, August 2017. In preparation.

[Kit95] A. Yu Kitaev. Quantum measurements and the Abelian stabilizer problem. *arXiv preprint quant-ph/9511026*, 1995.

[KSV02] Alexei Yu Kitaev, Alexander Shen, and Mikhail N. Vyalyi. *Classical and quantum computation*, volume 47. American Mathematical Society Providence, 2002.

[ME99] Michele Mosca and Artur Ekert. The hidden subgroup problem and eigenvalue estimation on a quantum computer. In *Quantum Computing and Quantum Communications*, pages 174–188. Springer, 1999.

[NIS16] NIST. *Post-Quantum Crypto Project*, December 16, 2016. http://csrc.nist.gov/groups/ST/post-quantum-crypto/.

[NSA16] NSA Information Assurance Directorate. *Commercial National Security Algorithm Suite and Quantum Computing FAQ*, January 2016. MFQ U/OO/815099-15.

[PJC+14] Shruti Patil, Ali JavadiAbhari, Chen-Fu Chiang, Jeff Heckey, Margaret Martonosi, and Frederic T. Chong. Characterizing the performance effect of trials and rotations in applications that use quantum phase estimation. In *Workload Characterization (IISWC), 2014 IEEE International Symposium on*, pages 181–190. IEEE, 2014.

[PZ03] John Proos and Christof Zalka. Shor's discrete logarithm quantum algorithm for elliptic curves. *Quantum Information & Computation*, 3(4):317–344, 2003.

[RNSL17] Martin Roetteler, Michael Naehrig, Krysta M. Svore, and Kristin Lauter. Quantum resource estimates for computing elliptic curve discrete logarithms. Cryptology ePrint Archive, Report 2017/598, 2017. `http://eprint.iacr.org/2017/598`.

[SHF14] Krysta M. Svore, Matthew B. Hastings, and Michael Freedman. Faster phase estimation. *Quantum Information & Computation*, 14(3-4):306–328, 2014.

[Sho99] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 41(2):303–332, 1999.

A Mosca and Ekert's algorithm

Mosca and Ekert's optimization [ME99] of Shor's algorithm [Sho99] provides the basis for the quantum magic box introduced in this paper. It is summarized here for background.

A.1 First stage

The first stage involves six main steps, each operating on two quantum registers, where the first register is an ℓ-qubit register and the second is large enough to hold a group element. The first stage involves a quantum circuit for applying the operator U_{a^x}, which multiplies the second register by a^x, where x is the value of the first register. The circuit can be optimized specifically for the base a, which is known in advance.

1. Initialize the first register to the ℓ-qubit state $|0\rangle$ and the second register to the group identity element $|1\rangle$:
$$|0\rangle\,|1\rangle\,.$$

2. Apply an ℓ-qubit Hadamard transform to the first register. This places the first register in the superposition of all possible ℓ-qubit states:
$$\frac{1}{2^{\ell/2}}\sum_{x=0}^{2^\ell-1}|x\rangle\,|1\rangle\,.$$

3. Apply the operator U_{a^x} to the second register, where x is the value of the first register. This places the two registers in the entangled superposition
$$\frac{1}{2^{\ell/2}}\sum_{x=0}^{2^\ell-1}|x\rangle\,U_{a^x}(|1\rangle) = \frac{1}{2^{\ell/2}}\sum_{x=0}^{2^\ell-1}|x\rangle\,|a^x\rangle\,.$$

4. Apply the Quantum Fourier Transform (QFT) to the first register. This results in the entangled superposition
$$\frac{1}{2^{\ell}}\sum_{y=0}^{2^\ell-1}|y\rangle\sum_{x=0}^{2^\ell-1}\exp(-2\pi iyx/2^\ell)\,|a^x\rangle\,.$$

5. Measure the first register to obtain a particular frequency y. The second register then collapses to the superposition $|\tilde{\Psi}_y\rangle$ defined as

$$|\tilde{\Psi}_y\rangle = \frac{1}{\sqrt{\tilde{A}_y}} \sum_{x=0}^{2^\ell - 1} \exp(-2\pi i y x / 2^\ell) \, |a^x\rangle \, ,$$

where $\tilde{A}_y$ is defined as

$$\tilde{A}_y = \left\| \sum_{x=0}^{2^\ell - 1} \exp(-2\pi i y x / 2^\ell) \, |a^x\rangle \right\|^2 .$$

The superposition $|\tilde{\Psi}_y\rangle$ may be considered an approximation to the eigenstate $|\Psi_k\rangle$ defined in Section 3. The probability of measuring y will be proportional to $\tilde{A}_y$.

6. Compute k by rounding $yr/2^\ell$ to the nearest integer, and output k. (This step may be performed classically).

The operator U_{a^x} can be implemented as a series of multiplications by successive squares of a, controlled by bits of the first register, as shown in Figure 2. In Mosca and Ekert's "flying qubits" optimization, the first register is emulated with a single qubit that is measured and reinitialized between each of the multiplications. The phase shift gates within the QFT are controlled "semi-classically" based on previous outputs. Thus, the number of qubits required by the first stage is at most one more than the size of a group element, plus any ancillae required by the multiplication operations.

A.2 Second stage

The second stage also involves six main steps, also operating on the same two registers. In the following, for simplicity, the second register is assumed to be in the superposition $|\Psi_k\rangle$ rather than its approximation. The approximation error can be made small enough with appropriate parameters that the result is not significantly affected. The second stage involves a quantum circuit for applying the operator U_{b^x}, which multiplies the second register by b^x. Similar to the first stage, the circuit can be optimized specifically for the target b.

1. Initialize the first register to the ℓ-qubit state $|0\rangle$. The second register continues in the superposition $|\Psi_k\rangle$:

$$|0\rangle \, |\Psi_k\rangle \, .$$

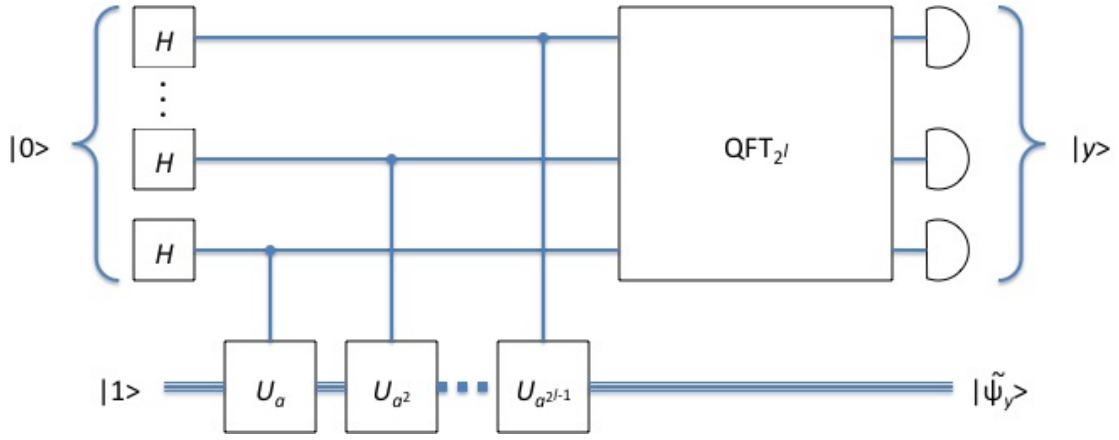

Figure 2: First stage of Mosca and Ekert's algorithm. H denotes a single-qubit Hadamard transform and $U_a, U_{a^2}, \ldots$ denote controlled multiplication by successive squares of a. The first register is measured at frequency $|y\rangle$; the second register continues in the approximate eigenstate $|\tilde{\Psi}_y\rangle$.

2. Apply an ℓ-qubit Hadamard transform to the first register. This again places the first register in a superposition of all possible ℓ-qubit states.

$$\frac{1}{2^{\ell/2}} \sum_{x=0}^{2^\ell - 1} |x\rangle \, |\Psi_k\rangle \, .$$

3. Apply the operator U_{b^x} to the second register, where x is the value of the first register. This places the two registers in the entangled superposition

$$\frac{1}{2^{\ell/2}} \sum_{x=0}^{2^\ell - 1} |x\rangle \, U_{b^x}(|\Psi_k\rangle) \, .$$

Because $|\Psi_k\rangle$ is an eigenstate of the operation U_a, it is also an eigenstate of U_b and of the repeated operation U_{b^x} for all x. Because $b^x = a^{mx}$, the effect of the operation U_{b^x} on the eigenstate $|\Psi_k\rangle$ is the same as a quantum phase shift of $\exp(2\pi ikmx/r)$. The resulting state is therefore equivalent to

$$\frac{1}{2^{\ell/2}} \sum_{x=0}^{2^\ell - 1} \exp(2\pi ikmx/r)|x\rangle \, |\Psi_k\rangle \, .$$

4. Apply the QFT to the first register. Because the dominant frequency in the phase in

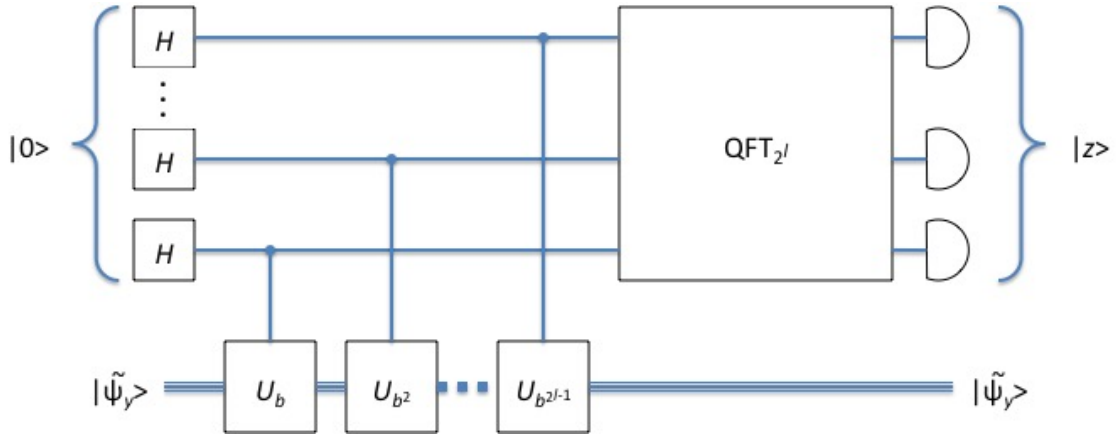

Figure 3: Second stage of Mosca and Ekert's algorithm. H denotes a single-qubit Hadamard transform and $U_b, U_{b^2}, \ldots$ denote controlled multiplication by successive squares of b. The first register is measured at frequency z, leading to the solution to the discrete logarithm problem.

the first register is km/r, this produces an entangled superposition that is approximately

$$|z\rangle\,|\Psi_k\rangle\,,$$

where $z = ((km \mod r)/r)2^\ell$. (This is an ℓ-bit estimate of the quantum phase shift angle θ.)

5. Measure the first register to obtain z with high probability.
6. Solve for the discrete logarithm m classically from k and z.

The operator U_{b^x} can be implemented as a series of multiplications by successive squares of b, similar to the first stage, as shown in Figure 3. Like the first stage, this stage can also be implemented with Mosca and Ekert's "flying qubits" optimization.