Shorのアルゴリズム実装動向調査

高安敦
情報通信研究機構　サイバーセキュリティ研究所　セキュリティ基盤研究室

2021年6月3日

目 次

第1章　エグゼクティブサマリー

Shor のアルゴリズムの実装・リソース評価の把握の重要性について．第 2 章で Shor のアルゴリズム [Sho94, Sho97] の実装・リソース評価の把握の重要性をまとめている．Shor のアルゴリズムは，多項式時間で素因数分解や離散対数問題を解けるために理論的には大きな脅威だが，実験的に暗号で用いるような大きなパラメータに対してこれらの問題を解いたという報告はこれまで行われていない．そのため，現状どの程度のパラメータまで実験的に適用可能なのか，実際に暗号で用いるような大きなパラメータの問題を解くにはどの程度の性能の量子コンピュータが必要なのかを把握しておくことは重要である．

Shor の量子アルゴリズムの解説．第 4 章において，Shor の量子アルゴリズムの解説を行う．それに先駆けて，第 3 で量子計算について説明する．ここでは，Shor のアルゴリズムを理解できるよう，古典ビットとは異なる量子ビットの概念，計算基底による測定，基本的な量子ゲートの紹介を行う．Shor の量子アルゴリズムを解説する第 4 章ではまず，合成数 $N = pq$ を素因数分解するためには，$\gcd(a, N) = 1$ を満たす整数 a に対して $a^r = 1 \mod N$ を満たす最小の正整数 r である位数を計算することで素因数分解を行えること示す．また，上記の位数計算とは類似の性質として，ある関数の周期を計算できれば離散対数問題を解けることを示す．次に，Shor のアルゴリズムの核となる量子フーリエ変換について説明する．最後に，冪乗剰余演算と量子フーリエ変換を用いることで，位数計算や周期計算を多項式時間で行う量子アルゴリズムが作れることを示す．これによって，素因数分解や素体・楕円曲線上の離散対数問題を多項式時間で解くことができる．

Shor の量子アルゴリズムについて報告されている実装結果の調査．第 5 章で，これまで確認されている Shor のアルゴリズムによる $N = 15$ または $N = 21$ の素因数分解 [ASK19, DLQ+20, LBC+12, LBYP07, MLLL+12, LWL+07, MNM+16, PMO09, SSV13, VSB+01] と $2^x = 1 \mod 3$ の素体上の離散対数問題 [AST+20] の実装結果を紹介する．基本的にほとんどの実験は $N = 15$ の素因数分解であるため，この章の内容を平易に理解できるように，まず第 4 章の復習として $N = 15$ を素因数分解するための量子アルゴリズムを詳細にまとめる．特に，$\gcd(a, 15) = 1$ を満たす $a \in \{2, 4, 7, 8, 11, 13, 14\}$ の位数は $r = 2$ または $r = 4$ のいずれかであり，[VSB+01] で考察されているように，実装の困難さは位数の値によって大きく異なる．そのため，位数が $r = 2$ または $r = 4$ の場合に分けて $N = 15$ を

素因数分解するための量子アルゴリズムを整理し，そのための量子回路を図 5.1–5.3 にまとめる．ただし，既存の Shor のアルゴリズムの実装実験においてはここで説明する一般的な量子回路が用いられることは稀で，ほとんど全ての論文で $N = 15$ や a または r の値に応じて効率化した量子回路を用いている．そのため，多くの論文で共通に用いられている効率化手法をまとめた後に，各実験で用いられている量子回路をまとめる．表 5.1 で各実験で用いた量子回路の量子ビット数や量子ゲートの数をまとめ，図 5.9–5.29 でそれらの量子回路を示す．ここで確認するように，ほとんどの結果は求めたい位数である $r = 2, 4$ の情報を用いて量子回路を設計するなど，一般的な合成数 $N = pq$ を素因数分解するときには適用できない効率化を行っている．さらに，[DLQ+20] は図 5.25 の量子回路を用いることで $N = 21$ の素因数分解には成功したが，図 5.28 の量子回路を用いて行った $N = 35$ の素因数分解には成功しなかったと報告している．これらの調査を踏まえ，2048 ビット合成数 $N = pq$ の素因数分解など暗号で用いられるような大きなパラメータの問題を解くには，量子コンピュータの物理的な飛躍が必要であり，実用的には現在用いられている公開鍵暗号方式の安全性が直ちに損なわれるわけではないことが期待される．

暗号で用いるようなパラメータに対して Shor の量子アルゴリズムを実行する際のリソース評価．第 6 章に量子アルゴリズムによって素因数分解や離散対数問題を解くためのリソース評価の既存研究をまとめた．表 6.4 において，現在知られている素因数分解と ECDLP 計算のための量子アルゴリズムの漸近的なリソース評価，表 6.5, 6.6 において暗号で用いられるパラメータに対するリソース評価をまとめている．これまでの研究で量子アルゴリズム実装が大幅に進歩しており，最新の Gidney と Ekerå による素因数分解 [GE19] や Häner ら [HJN+20] と Banegas ら [BBvHL20] による ECDLP 計算は従来の結果を大幅に効率化している．そのため，今後もこの分野の発展によって素因数分解や離散対数問題計算のための量子アルゴリズムがより効率化する可能性があるため，注意が必要である．これらには劣るが，Gheorghiu と Mosca [GM19] はゲートエラー率が 10^{-3} と 10^{-5} の場合に素因数分解と ECDLP 計算に必要な量子ビット数と計算時間のトレードオフを解析しており，その結果を図 6.2–6.12 にまとめている．また，前述の最も効率的な素因数分解アルゴリズムである Gidney と Ekerå [GE19] が用いた効率化技法をまとめ，この技法による素因数分解と素体上の離散対数計算のためのリソース評価を表 6.9–6.14 にまとめている．

第2章　はじめに

現在実用的に用いられている公開鍵暗号方式である RSA 暗号 [RSA78],（楕円曲線上の）Diffie-Hellman 鍵共有プロトコル [DH76],（EC)DSA [NIST13] の安全性は，素因数分解や（楕円曲線上の）離散対数問題の計算量的困難性と密接な関わりがある．これまでこれらの問題を解くアルゴリズムを高速化する多くの研究 [LJMP90] が行われてきたが，いずれの問題に対しても多項式時間アルゴリズムは知られていない．これまで実験的に解かれた最も大きなパラメータは，829 ビット合成数の素因数分解と 795 ビット素体上の離散対数問題 [BGG+20]，約 117 ビット楕円曲線上の離散対数問題 [BEL+16] であり，依然実用的に用いられているパラメータとは開きがある．そのため，上記の公開鍵暗号方式は安全だと信じられている．

上記の結果は現在普及している古典計算機に限定したときの話であり，Shor は量子コンピュータを用いれば素因数分解や離散対数問題を多項式時間で解けることを示した [Sho94, Sho97]．それゆえ，現在暗号で用いられたパラメータを破ることができる実用的な量子コンピュータが完成すれば，現在用いられている公開鍵暗号方式は容易に破られてしまう．そのため，量子コンピュータによっても解読が計算量的に困難な耐量子計算機暗号方式の開発が活発に研究されている．さらに，2017 年に始まった NIST による耐量子計算機暗号方式の標準化計画によりこの流れは加速しており，現在用いられている公開鍵暗号方式から耐量子計算機暗号方式へ今後移行されるであろうことが予想される．だが，これまで報告されている量子コンピュータはいずれも小規模なもので，現在暗号で用いられたパラメータを破ることは困難であると予想される．

そのため，本報告書では実用的な Shor のアルゴリズムの脅威に関する調査を行う．まず，これまでの Shor のアルゴリズムの実験に関する調査を行う．これまで，$15, 21$ という非常に小さな合成数の素因数分解と $2^x = 1 \mod 3$ という非常に小さなパラメータにおける離散対数問題の実験しか確認されず，2048 ビットなど暗号で用いるパラメータとは大きなギャップがある．また，これらの実験で用いられた量子回路は小さな合成数に特化したいくつかの効率化が行われており，同様の手法を一般の合成数に適用するのは困難だと考えられる．量子コンピュータによる実験によって素因数分解されたのが $15, 21$ のような小さな合成数であるというのは，Shor のアルゴリズムに限定した話である．実際，量子アニーリングによって $376289 = 571 \times 659$ の素因数分解 [JBM+18] や，Grover のアルゴリ

ズム [Gro96] を用いた $4088459 = 2017 \times 2027$ の素因数分解 [DSBP18] などが報告されている．ただし，これらのアルゴリズムは Shor のアルゴリズムのように多項式時間で動くという保証がない．言い換えると，数体篩法などの古典アルゴリズムよりも効率的であるという保証がない．そのため，今回の調査では Shor のアルゴリズム以外の量子アルゴリズムは対象としない．このように，既存の実験結果のみから判断すれば Shor のアルゴリズムによって現在用いられている公開鍵暗号方式が破られるのはかなり先になると期待される．ところが，量子コンピュータの開発も活発に研究されているテーマであり，数年後や数十年後にどれだけ大規模な量子コンピュータが完成しているのかを予想するのは難しい．そのため，いずれ大規模量子コンピュータが完成する未来に備えて，Shor のアルゴリズムを実装する際のリソース評価を行い，どれだけのスペックの量子コンピュータが完成すれば現在用いられている暗号が破れうるのかを見積もる．この調査によって，量子コンピュータ開発分野の発展がどれほど暗号の安全性に影響を及ぼすのかを精密に見積もることができる．

本報告書の構成を説明する．第 3 章で，量子計算の基礎について説明する．古典計算とは異なる量子ビットや測定の概念を説明し，量子回路モデルや量子ゲートについて説明する．第 4 章で，Shor のアルゴリズムを説明する．Shor のアルゴリズムの核となる量子フーリエ変換を説明し，Shor のアルゴリズムの量子パートに相当する位数計算アルゴリズムや，周期計算アルゴリズムの説明を行う．第 5 章で，Shor のアルゴリズムの既存の実装結果をまとめる．前述のように，これまで実験的に素因数分解されたのは $N = 15, 21$ などの小さな合成数や素体上の離散対数問題のみである．これらの実験の量子回路を理解しやすくするために，$N = 15$ のときの Shor のアルゴリズムの復習を行なった後，実験を成功するために用いられた効率化手手法をまとめ，各実験で用いられた量子回路をまとめる．第 6 章で，Shor のアルゴリズムを実行する際のリソース評価に関する既存結果をまとめる．

第3章　量子計算

古典計算では，$\{0,1\}$ で表される古典ビットを用いて議論する．量子コンピュータでも同様に量子ビット (qubit) を用いて議論を進めるが，その概念は古典ビットとは大きく異なる．本章では，量子ビット系とそれを用いた量子計算の基礎について簡単にまとめる．本章の内容は [IOK+12, NC11] を参考にしている．より詳しくはこれらの文献を参照されたい．

3.1　量子ビット

古典ビットでは，1 ビットの状態は “0” と “1” いずれかによって表される．同様に，量子ビットでは，1 量子ビット状態として “$|0\rangle$” と “$|1\rangle$” があり，それぞれ 2 次元ベクトル

$$|0\rangle \coloneqq \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad |1\rangle \coloneqq \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

を表す．ただし，古典ビットのときとは違い，1 量子ビットは $|0\rangle$ と $|1\rangle$ 以外の状態になりうる．より正確には，1 量子ビットは $|0\rangle$ と $|1\rangle$ の線型結合で表される状態

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

を取ることができ，これを重ね合わせ (superposition) 状態と呼ぶ．ここで，線型結合の係数は $\alpha, \beta \in \mathbb{C}$ かつ $|\alpha|^2 + |\beta|^2 = 1$ を満たす．つまり，1 量子ビット状態は $\mathbb{C}^2$ 上の単位ベクトルで表される．1 量子ビット状態を記述する際に，正規直行基底 $\{|0\rangle, |1\rangle\}$ は特に計算基底 (computational basis) と呼ばれる．

古典的な 2 ビットは，00, 01, 10, 11 と 4 つの状態を持つ．同様に，2 量子ビット系の計算基底は 4 つの 2 量子ビット状態 $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ を持つ．2 量子ビット系を複素ベクトルとして見るとき，$j_1, j_2 \in \{0,1\}$ となる $|j_1 j_2\rangle$ は $|j_1\rangle$ と $|j_2\rangle$ のテンソル積 $|j_1\rangle \otimes |j_2\rangle$ で表される 4 次元複素ベクトルで

あり，

$$|00\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \qquad |01\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \qquad |10\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \qquad |11\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

となる．1 量子ビット系と同様に，2 量子ビット系の量子状態を計算基底 $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ の複素数係数線型結合によって

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle = \begin{pmatrix} \alpha_{00} \\ \alpha_{01} \\ \alpha_{10} \\ \alpha_{11} \end{pmatrix}$$

と書くことができる．ただし，線型結合の係数は $|\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 = 1$ を満たす．つまり，2 量子ビット状態は $\mathbb{C}^4$ 上の単位ベクトルで表される．2 量子ビット $|\psi\rangle$ は，二つの 1 量子ビット $|\phi_0\rangle$ と $|\phi_1\rangle$ を用いて常に $|\psi\rangle = |\phi_0\rangle \otimes |\phi_1\rangle$ と書けるわけではないことに注意されたい．このとき，$|\psi\rangle$ はエンタングル状態 (量子もつれ状態，量子絡み合い状態) であるという．逆に，$|\psi\rangle = |\phi_0\rangle \otimes |\phi_1\rangle$ と書けるときには $|\psi\rangle$ は積状態であるという．

一般の n 量子ビット系についてもこれまでと同様である．n 量子ビット系の計算基底は 2^n 個の n 量子ビット状態 $\{|j_1 j_2 \cdots j_n\rangle\}_{j_1,j_2,\ldots,j_n \in \{0,1\}}$ を持つ．n 量子ビット系を複素ベクトルとして見るとき，$j_1, j_2, \ldots, j_n \in \{0,1\}$ となる $|j_1 j_2 \cdots j_n\rangle$ は $|j_1\rangle, |j_2\rangle, \ldots, |j_n\rangle$ のテンソル積 $|j_1\rangle \otimes |j_2\rangle \otimes \cdots \otimes |j_n\rangle$ で表される 2^n 次元複素ベクトルである．n 量子ビット系の量子状態を計算基底 $\{|j_1 j_2 \cdots j_n\rangle\}_{j_1,j_2,\ldots,j_n \in \{0,1\}}$ の複素数係数線型結合によって

$$|\psi\rangle = \sum_{j_1,j_2,\ldots,j_n \in \{0,1\}} \alpha_{j_1 j_2 \cdots j_n} |j_1 j_2 \cdots j_n\rangle$$

と書くことができ，$\sum_{j_1,j_2,\ldots,j_n \in \{0,1\}} |\alpha_{j_1 j_2 \cdots j_n}|^2 = 1$ を満たす．つまり，n 量子ビット状態は $\mathbb{C}^{2^n}$ 上の単位ベクトルで表される．

3.2 測定

古典計算において，1 ビット状態 $j \in \{0,1\}$ を測定すれば，j として 0 または 1 いずれかの値を観測することができる．量子ビット系では，1 量子ビットが $\{|0\rangle, |1\rangle\}$ のみならず，それらの重ね合わせ状

態 $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ を取ることができるため，1 量子ビットは線型結合係数 (α, β) によって 1 古典ビットより多くの情報を取ることができるのは先ほど述べた．ただし，1 量子ビット状態 $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ を測定しても $|0\rangle$ または $|1\rangle$ いずれかの値しか観測することはできず，線型結合係数 (α, β) の情報を陽に得ることはできない．その代わり線型結合係数 (α, β) は，1 量子ビット状態 $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ を測定したときに $|0\rangle$ または $|1\rangle$ いずれの値を観測するかの確率を与える．より正確には，量子ビット $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ を計算基底[1]$\{|0\rangle, |1\rangle\}$ で測定したとき，$|j\rangle \in \{|0\rangle, |1\rangle\}$ を観測する確率は，観測結果となる量子状態 $|j\rangle$ と測定される量子状態 $|\psi\rangle$ の内積の 2 乗

$$|\langle j|\psi\rangle|^2 \tag{3.1}$$

で表される．ここで，列ベクトル $|\psi\rangle = (\alpha, \beta)^T \in \mathbb{C}^2$ に対して $\langle\psi|$ は複素共役な行ベクトル $\langle\psi| = (\bar{\alpha}, \bar{\beta})$ を表す．具体的には，量子ビット $|\psi\rangle$ を測定して $|0\rangle$ を観測する確率は

$$|\langle 0|\psi\rangle|^2 = \left|(1,0)\begin{pmatrix}\alpha\\ \beta\end{pmatrix}\right|^2 = |\alpha|^2$$

であり，同様の計算により，$|1\rangle$ を観測する確率は $|\beta|^2$ となる．ここで，1 量子ビット状態 $|\psi\rangle$ は $\mathbb{C}^2$ 上の単位ベクトルであったので，$|\alpha|^2 + |\beta|^2 = 1$ で，計算基底 $\{|0\rangle, |1\rangle\}$ による測定で $|0\rangle$ または $|1\rangle$ いずれかの値を必ず観測できることに注意されたい．この性質により，線型結合係数 (α, β) は振幅 (amplitude) と呼ばれる．また，$|\alpha|^2 + |\beta|^2 = 1$ という条件は，振幅を正規化するものと考えることができる．

2 量子ビット状態 $|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$ においても，振幅が条件 $|\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 = 1$ によって正規化されており，この 2 量子ビット状態を計算基底 $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ で測定したときに $|j_1 j_2\rangle$ を観測する確率は，式 (3.1) を一般化し

$$|\langle j_1 j_2|\psi\rangle|^2 = |\alpha_{j_1 j_2}|^2 \tag{3.2}$$

となる．これは，一般の n 量子ビット状態でも同様である．つまり，n 量子ビット状態 $|\psi\rangle = \sum_{j_1, j_2, \ldots, j_n \in \{0,1\}} \alpha_{j_1 j_2 \cdots j_n} |j_1 j_2 \cdots j_n\rangle$ を計算基底 $\{|j_1 j_2 \cdots j_n\rangle\}_{j_1, j_2, \ldots, j_n \in \{0,1\}}$ で測定したときに $|j_1 j_2 \cdots j_n\rangle$ を観測する確率は $|\langle j_1 j_2 \cdots j_n|\psi\rangle|^2 = |\alpha_{j_1 j_2 \ldots j_n}|^2$ となる．

ただし，多量子ビット状態では，全ての状態ではなく部分的な状態の測定を行うことができる．2 量子ビット状態 $|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$ の第 1 量子ビットを計算基底 $\{|0\rangle, |1\rangle\}$

[1] 一般的には，量子状態の測定は任意の正規直行基底 $\{|\phi_0\rangle, |\phi_1\rangle\}$ で行うことができるが，ここでは簡単のために計算基底 $\{|0\rangle, |1\rangle\}$ による測定のみを考える．ただし，正規直行基底 $\{|\phi_0\rangle, |\phi_1\rangle\}$ による測定で $|\phi_0\rangle$ または $|\phi_1\rangle$ を観測する確率は，$|j\rangle \in \{|\phi_0\rangle, |\phi_1\rangle\}$ として式 (3.1) によって与えられる．

で測定すると，確率 $|\alpha_{00}|^2 + |\alpha_{01}|^2$ で $|0\rangle$ を観測し，その後の量子状態は

$$|\psi'\rangle = \frac{\alpha_{00}|00\rangle + \alpha_{01}|01\rangle}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}}$$

となる．つまり，第 1 量子ビットとして $|0\rangle$ を観測すると，第 1 量子ビットが $|1\rangle$ である状態 $|10\rangle, |11\rangle$ の情報は失われる．このときの観測確率は，観測結果となる量子状態 $|0\rangle$ と測定される量子状態 $|\psi\rangle$ の内積を部分的に取ったとき

$$\begin{aligned}
\langle 0|\psi\rangle &= \alpha_{00}\langle 0|00\rangle + \alpha_{01}\langle 0|01\rangle + \alpha_{10}\langle 0|10\rangle + \alpha_{11}\langle 0|11\rangle \\
&= \alpha_{00}\langle 0|0\rangle \otimes |0\rangle + \alpha_{01}\langle 0|0\rangle \otimes |1\rangle + \alpha_{10}\langle 0|1\rangle \otimes |0\rangle + \alpha_{11}\langle 0|1\rangle \otimes |1\rangle \\
&= \alpha_{00}|0\rangle + \alpha_{01}|1\rangle \\
&= \begin{pmatrix} \alpha_{00} \\ \alpha_{01} \end{pmatrix}
\end{aligned}$$

と書けることに注意し，

$$\|\langle 0|\psi\rangle\|^2$$

で表される．観測後の量子状態は，振幅を正規化して $\langle 0|\psi\rangle / \|\langle 0|\psi\rangle\|$ と書くことができる．

一般の n 量子ビット系においても同様である．つまり，n 量子ビット状態 $|\psi\rangle = \sum_{j_1,j_2,\ldots,j_n\in\{0,1\}} \alpha_{j_1 j_2 \cdots j_n}|j_1 j_2 \cdots j_n\rangle$ の最初の m 量子ビットを計算基底 $\{|i_1 i_2 \cdots i_m\rangle\}_{i_1,i_2,\ldots,i_m\in\{0,1\}}$ で測定したときに $|i_1 i_2 \cdots i_m\rangle$ を観測する確率は

$$\|\langle i_1 i_2 \cdots i_m|\psi\rangle\|^2 = \sum_{j_{m+1},\ldots,j_n\in\{0,1\}} |\alpha_{i_1 \ldots i_m j_{m+1} \ldots j_n}|^2$$

となり，測定後の量子状態は $\langle i_1 i_2 \cdots i_m|\psi\rangle / \|\langle i_1 i_2 \cdots i_m|\psi\rangle\|$ となる．

3.3 量子ゲート

本報告書では，量子計算の記述に量子回路モデルを用いる．量子回路はワイヤと量子ゲートからなり，ワイヤは情報の伝達を行い，量子ゲートは入力ワイヤの情報に演算を施しその結果を出力ワイヤに送る．量子計算における制約で，量子ゲート演算はユニタリ変換でなければならない．ユニタリ行列とは，$UU^\dagger = U^\dagger U = I$ となる行列 U のことであり，量子ゲートはユニタリ行列で記述することが

できる．つまり，行列 U で記述される量子ゲートは，行列 $U^\dagger$ で記述される量子ゲートがあるという意味で可逆的である．ここで，行列 $U^\dagger$ は行列 U の共役転置を表し，行列 I は単位行列である．

簡単な例を用いて量子ゲート演算について説明する．古典回路モデルでは $\neg$ は NOT 演算子を表し，$\neg 0 = 1, \neg 1 = 0$ である．量子計算でこれに対応するユニタリ変換は次の行列

$$X \coloneqq \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

であり，以下の計算によって確認できる．

$$X|0\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle, \qquad X|1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle$$

つまり，1 量子ビット状態 $|0\rangle$ と $|1\rangle$ をそれぞれ $|1\rangle$ と $|0\rangle$ に反転させることができる．ただし，重ね合わせ状態 $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ を入力としたときは，

$$X|\psi\rangle = \begin{pmatrix} \beta \\ \alpha \end{pmatrix}$$

と 2 次元ベクトルの成分が反転することを注意されたい．入力量子ビットに量子ゲート X を作用させて測定を行う量子回路を図 3.1 のように書く．慣習に倣い，量子状態は左から右に時間発展することにする．

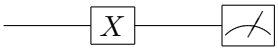

図 3.1: 量子回路の例

量子ゲートがユニタリ行列で表現されていることは，入出力ワイヤの本数は等しいことを意味する．つまり，古典回路の $\wedge$ や $\vee$ のような 2 入力 1 出力な量子ゲートは存在しない．量子ゲートがユニタリ行列であるという制約は，量子ゲート演算による入力状態を $|\psi\rangle$, 量子ゲートを U, 出力状態を $|\psi'\rangle = U|\psi\rangle$ と書いたとき，

$$\langle\psi'|\psi'\rangle = \langle\psi|U^\dagger U|\psi\rangle = \langle\psi|\psi\rangle = 1$$

となり，振幅の 2 乗和が 1 になるという条件が保存されることに起因する．

これまでの議論は，多量子ビット系においても同様である．また，測定のときと同様，多量子ビットの一部の量子ビットのみに量子ゲートを適用することも可能である．例えば，行列

$$I \otimes X = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

は，以下の計算

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha_{00} \\ \alpha_{01} \\ \alpha_{10} \\ \alpha_{11} \end{pmatrix} = \begin{pmatrix} \alpha_{01} \\ \alpha_{00} \\ \alpha_{11} \\ \alpha_{10} \end{pmatrix}$$

より，

$$I \otimes X(\alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle) = \alpha_{01}|00\rangle + \alpha_{00}|01\rangle + \alpha_{11}|10\rangle + \alpha_{10}|11\rangle$$

を満たすので，入力の 2 量子ビット系 $\alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$ の第 2 ビットのみを反転させている．逆に，行列

$$X \otimes I = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

は，以下の計算

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} \alpha_{00} \\ \alpha_{01} \\ \alpha_{10} \\ \alpha_{11} \end{pmatrix} = \begin{pmatrix} \alpha_{10} \\ \alpha_{11} \\ \alpha_{00} \\ \alpha_{01} \end{pmatrix}$$

より，

$$X \otimes I(\alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle) = \alpha_{10}|00\rangle + \alpha_{11}|01\rangle + \alpha_{00}|10\rangle + \alpha_{01}|11\rangle$$

を満たすので，入力の 2 量子ビット系 $\alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$ の第 1 ビットのみを反転させている．これを一般化し，行列 $I^{\otimes m-1} \otimes X \otimes I^{\otimes n-m}$ は，n 量子ビット状態 $\sum_{j_1,j_2,\ldots,j_n\in\{0,1\}} \alpha_{j_1j_2\cdots j_n}|j_1j_2\cdots j_n\rangle$ の左から m ビット目のみを反転させ，

$$
\begin{aligned}
&I^{\otimes m-1} \otimes X \otimes I^{\otimes n-m} \sum_{j_1,j_2,\ldots,j_n\in\{0,1\}} \alpha_{j_1j_2\cdots j_n}|j_1j_2\cdots j_n\rangle \\
&= \sum_{j_1,j_2,\ldots,j_n\in\{0,1\}} \alpha_{j_1\cdots j_{m-1}\neg j_m j_{m+1}\cdots j_n}|j_1j_2\cdots j_n\rangle
\end{aligned}
$$

となる．その他にも，行列 $U \in \mathbb{C}^{2^n\times 2^n}$ がユニタリ変換であるならば n ビットの量子ゲートとなる．入力量子ビットに 2 ビット量子ゲート U を作用させて測定を行う量子回路を図 3.2 のように書く．これは一般の n ビットゲートの場合でも同様である．

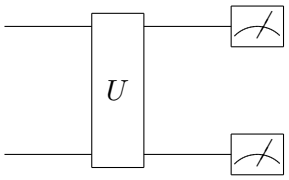

図 3.2: 2 ビットゲートを持つ量子回路の例

第 3.3.1–3.3.6 小節で，本報告書で扱ういくつかの代表的な量子ゲートをまとめる．

3.3.1 Pauli ゲート

前述の行列 X に加え，以下の行列 Y, Z を合わせた 3 つの行列で表される量子ゲートを Pauli ゲートと呼ぶ．

$$
X \coloneqq \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad Y \coloneqq \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \qquad Z \coloneqq \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}
$$

3.3.2 Hadamard ゲート

行列

$$H \coloneqq \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

で表される量子ゲートを Hadamard ゲートと呼び，多くの量子アルゴリズムで利用されている．重要な性質として，

$$H|0\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \qquad H|1\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \frac{|0\rangle - |1\rangle}{\sqrt{2}} \tag{3.3}$$

という関係が成り立つ．これは，初期状態 $|0\rangle$ を用意すれば，Hadamard 変換により $|0\rangle$ と $|1\rangle$ の振幅が等しい 1 量子状態を作れることを意味している．より一般的には，量子ゲート $H^{\otimes n}$ に n 量子ビット状態 $|00\cdots0\rangle$ を入力すると，

$$H^{\otimes n}|00\cdots0\rangle = \frac{1}{\sqrt{2^n}} \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2^n}} \sum_{j_1,j_2,\ldots,j_n \in \{0,1\}} |j_1 j_2 \cdots j_n\rangle$$

より，n 量子ビット系の全ての計算基底 $\{|j_1 j_2 \cdots j_n\rangle\}_{j_1,j_2,\ldots,j_n \in \{0,1\}}$ の振幅が等しい量子状態を作ることができる．この計算が正しいことは，$2^n \times 2^n$ 行列 $H^{\otimes n}$ の第 1 列の要素は全て 1 で，$|00\cdots0\rangle$ は第 1 成分のみ 1 で他は全て 0 のベクトルであることから容易に確認できる．

3.3.3 回転ゲート

一般に，以下の行列

$$R_k \coloneqq \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix}$$

によって表される量子ゲートを回転ゲートと呼ぶことにする．回転ゲートは，量子状態 $(|0\rangle + e^{2\pi i\theta}|1\rangle)/\sqrt{2}$ を入力としたとき，

$$\begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ e^{2\pi i\theta} \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ e^{2\pi i(\theta+1/2^k)} \end{pmatrix}$$

$$= \frac{|0\rangle + e^{2\pi i(\theta+1/2^k)}|1\rangle}{\sqrt{2}}$$

より，$|1\rangle$ の係数を角度 $2\pi/2^k$ だけ回転させる作用を持つ．特に，$k = 2, 4$ のときの行列 R_k を次のように書く．

$$S := R_2 = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$$
$$T := R_4 = \begin{pmatrix} 1 & 0 \\ 0 & e^{\pi i/4} \end{pmatrix}$$

3.3.4 制御 NOT ゲート

これまで 1 ビットゲートを見てきたが，次の行列 $\mathsf{CNOT} \in \mathbb{C}^{4\times4}$ によって表される量子ゲートを制御 NOT ゲートと呼ぶ．

$$\mathsf{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

制御 NOT ゲートは，

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \qquad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

より，第 1 ビットが $|0\rangle$ である 2 量子ビット状態 $|00\rangle, |01\rangle$ を変化させない．逆に，

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \qquad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

より，第 1 ビットが $|1\rangle$ である 2 量子ビット状態 $|00\rangle, |01\rangle$ の第 1 ビットを変化させずに第 2 ビットのみを反転させる．つまり，2 量子ビット $|i_1\rangle|i_2\rangle$ を $|i_1\rangle|i_1 \oplus i_2\rangle$ と変化させると書くことができる．こ

のとき，第 1 ビットを制御ビットと呼び，第 2 ビットを標的ビットと呼ぶ．第 1 ビットを制御ビットとし，第 2 ビットを標的ビットとする制御 NOT ゲートの量子回路を図 3.3 のように書く．2 つのワイヤのうち，上が制御ビットとなる第 1 ビットで下が標的ビットとなる第 2 ビットである．

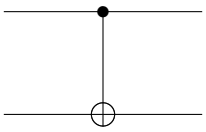

図 3.3: 量子回路における制御 NOT ゲート

制御 NOT ゲートは，

$$\begin{aligned}\mathsf{CNOT} &= \begin{pmatrix} I & O \\ O & X \end{pmatrix} \\ &= \begin{pmatrix} I & O \\ O & O \end{pmatrix} + \begin{pmatrix} O & O \\ O & X \end{pmatrix} \\ &= \langle 0|0\rangle \otimes I + \langle 1|1\rangle \otimes X\end{aligned}$$

と分解することができ，一般に，行列

$$\begin{aligned}C\text{-}U &= \begin{pmatrix} I & O \\ O & U \end{pmatrix} \\ &= \begin{pmatrix} I & O \\ O & O \end{pmatrix} + \begin{pmatrix} O & O \\ O & U \end{pmatrix} \\ &= \langle 0|0\rangle \otimes I + \langle 1|1\rangle \otimes U\end{aligned}$$

で表される量子ゲートは，$|0\rangle|i\rangle$ の量子状態を変化させず，$|1\rangle|i\rangle$ を $|1\rangle \otimes U|i\rangle$ に変化させる制御演算を行うことができる．第 1 ビットを制御ビットとして第 2 ビットに U ゲートを作用させる量子回路を図 3.4 のように書く．

3.3.5　スワップゲート

2 ビットのスワップ操作は，第 1 ビットと第 2 ビットを入れ替えるものである．スワップゲートを量子回路では図 3.5 のように書く．

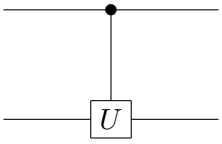

図 3.4: 量子回路における C-U ゲート

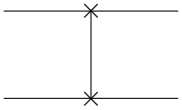

図 3.5: 量子回路におけるスワップゲート

ただし，スワップゲートは基本的には図 3.6 のように三つの制御 NOT ゲートを組み合わせて実現される．

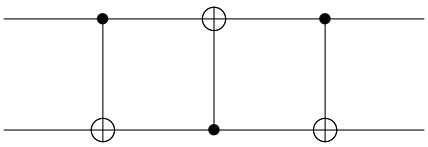

図 3.6: 制御 NOT ゲートによるスワップ

実際に，図 3.6 の量子回路でスワップが行われていることを確認する．入力量子状態を

$$\alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$$

とする．第 1 ビットを制御ビットとする最初の制御 NOT ゲートを作用させると，量子状態は

$$\alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|11\rangle + \alpha_{11}|10\rangle$$

と変化する．次に，第 2 ビットを制御ビットとする制御 NOT ゲートを作用させると，量子状態は

$$\alpha_{00}|00\rangle + \alpha_{01}|11\rangle + \alpha_{10}|01\rangle + \alpha_{11}|10\rangle$$

となり，最後の制御 NOT ゲートによって量子状態は

$$\alpha_{00}|00\rangle + \alpha_{01}|10\rangle + \alpha_{10}|01\rangle + \alpha_{11}|11\rangle$$

となり，スワップが行われたことが確認できる．

3.3.6 Toffoli ゲート

次の行列

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

で表される 3 ビット量子ゲートを Toffoli ゲートと呼ぶ．Toffoli ゲートは，3 量子ビットのうち二つの制御ビットを持つ制御演算を行う量子ゲートである．上の行列の場合，最初の 2 量子ビットを制御ビットとし，制御ビットがいずれも 1 の場合のみ第 3 ビットを反転させる．つまり，$|000\rangle, |001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle$ を入力としたときは量子ビットを変化させず，$|110\rangle$ と $|111\rangle$ を入力とした場合にはそれぞれ $|111\rangle$ と $|110\rangle$ を出力する．量子回路において，Toffoli ゲートは図 3.7 で表される．

図 3.8 は，$T, T^{\dagger}$, 制御 NOT ゲートによって分解した Toffoli ゲートである．

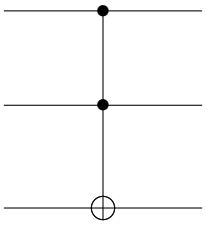

図 3.7: 量子回路における Toffoli ゲート

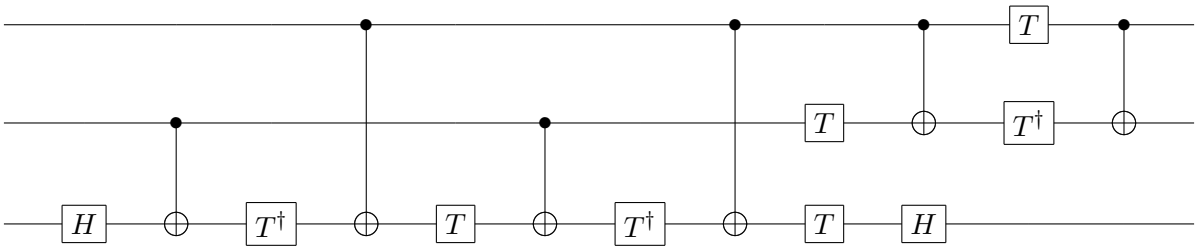

図 3.8: 量子回路における $T, T^\dagger$, 制御 NOT ゲートで分解された Toffoli ゲート

第4章　Shorのアルゴリズム

この章で，Shor のアルゴリズムを説明する．第 4.1 節で，Shor のアルゴリズムの概要を説明する．ただし，ここでは量子アルゴリズムの詳細には触れない．第 4.2 節で，Shor のアルゴリズムの核となる量子フーリエ変換を説明する．第 4.3 節で，Shor の素因数分解の量子パートとなる位数計算アルゴリズムを説明する．第 4.4 節で，周期計算アルゴリズムとそれを用いた離散対数問題アルゴリズムを説明する．この章の内容の多くは [NC11] を参考にしている．

4.1　Shor のアルゴリズムの概要

Shor のアルゴリズムによる n ビットの合成数 $N = pq$ の素因数分解は以下の操作で行われる．

ステップ 1:　整数 $a \leftarrow \{2, 3, \ldots, N-1\}$ をサンプルし，$\gcd(a, N) \neq 1$ ならば $\gcd(a, N)$ を出力する．そうでないならばステップ 2 に進む．

ステップ 2:　$a^r = 1 \mod N$ を満たす最小の正整数 r を計算する．

ステップ 3:　r が奇数のとき，ステップ 1 に戻る．r が偶数ならば $(a^{r/2}+1)(a^{r/2}-1) = 0 \mod N$ が成り立つので，$\gcd(a^{2/r}+1, N)$ または $\gcd(a^{2/r}-1, N)$ を出力する．

ステップ 3 において r は $1/2$ より大きな確率で偶数となるので，有意な確率で Shor のアルゴリズムは合成数 N の素因数分解に成功する．

ステップ 1 とステップ 3 は古典的に容易に計算できるが，量子アルゴリズムを用いるのはステップ 2 において a の位数計算を行うところである．このとき，関数

$$f_N(x) \coloneqq a^x \mod N$$

における位数 r は，ステップ 2 の条件により全ての $x \in \{1, 2, \ldots, N\}$ に対して

$$f_N(x+r) = a^{x+r} \mod N$$

$$
\begin{aligned}
&= a^x \cdot a^r \mod N \\
&= a^x \mod N \\
&= f(x)
\end{aligned}
$$

を満たす最小の正整数である．位数計算のための量子アルゴリズムについては第 4.3 節でまとめる．

離散対数問題も同様に位数計算アルゴリズムによって解くことができる．N, g, h が与えられたときに，$h = g^s \mod N$ を満たす離散対数 s を求めたい．ここで，関数

$$
\begin{aligned}
f_{g,h}(x_1, x_2) &:= h^{x_1} \cdot g^{x_2} \mod N \\
&= g^{sx_1 + x_2} \mod N
\end{aligned}
$$

を考える．このとき，関数 $f_{g,h}(x, y)$ は次のように位数 $(1, -s)$ を持つことがわかる．

$$
\begin{aligned}
f_{g,h}(x_1 + 1, x_2 - s) &= g^{s(x_1+1)+x_2-s} \mod N \\
&= g^{sx_1+x_2} \mod N \\
&= f_{g,h}(x_1, x_2)
\end{aligned}
$$

よって，素因数分解と類似の手法で離散対数問題を解くことができる．

4.2 量子フーリエ変換

第 4.3 節の位数計算アルゴリズムを説明する前に，この節では位数計算アルゴリズムにおいて用いる量子フーリエ変換について説明する．

M 本の正規直交基底 $|0\rangle, \ldots, |M\rangle$ における状態 $|j\rangle$ に対する量子フーリエ変換は次の操作のことである．

$$
|j\rangle \to \frac{1}{\sqrt{M}} \sum_{k=0}^{M-1} e^{2\pi i j k / M} |k\rangle \tag{4.1}
$$

さらに，任意の状態 $\sum_{j=0}^{M-1} x_j |j\rangle$ に対しては次のように書くことができる．

$$
\sum_{j=0}^{M-1} x_j |j\rangle \to \sum_{k=0}^{M-1} y_k |k\rangle
$$

ただし，係数 y_k は次のものである．

$$y_k \coloneqq \frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} x_j e^{2\pi i j k/M}$$

定義からは明らかではないが，この操作はユニタリであり量子計算によって記述可能である．ここで，$M = 2^m$ として $j = j_{m-1}2^{m-1} + j_{m-2}2^{m-2} + j_0$ に対して j を $j_{m-1}j_{m-2}\cdots j_0$ と書くことにする．同様に，$j_\ell/2 + j_{\ell+1}/4 + \cdots + j_n/2^{n-\ell+1}$ を $0.j_\ell j_{\ell+1}\cdots j_{n-\ell+1}$ と書く．このとき，式 (4.1) を次のように変形することができる．

$$\begin{aligned}
|j\rangle &\to \frac{1}{\sqrt{2^m}} \sum_{k=0}^{2^m-1} e^{2\pi i j k/2^m} |k\rangle \\
&= \frac{1}{\sqrt{2^m}} \sum_{k_{m-1}\in\{0,1\}} \cdots \sum_{k_0\in\{0,1\}} e^{2\pi i j (k_{m-1}2^{m-1} + k_{m-2}2^{m-2} + \cdots + k_0 2^0)/2^m} |k_{m-1}\cdots k_0\rangle \\
&= \frac{1}{\sqrt{2^m}} \sum_{k_{m-1}\in\{0,1\}} \cdots \sum_{k_0\in\{0,1\}} \bigotimes_{\ell=1}^{m} e^{2\pi i j k_{m-\ell}/2^\ell} |k_{m-\ell}\rangle \\
&= \frac{1}{\sqrt{2^m}} \bigotimes_{\ell=1}^{m} \left(\sum_{k_{m-\ell}\in\{0,1\}} e^{2\pi i j k_{m-\ell}/2^\ell} |k_{m-\ell}\rangle \right) \\
&= \frac{1}{\sqrt{2^m}} \bigotimes_{\ell=1}^{m} \left(|0\rangle + e^{2\pi i j/2^\ell} |1\rangle \right) \\
&= \frac{\left(|0\rangle + e^{2\pi i \cdot 0.j_{m-1}j_{m-2}\cdots j_0}|1\rangle\right) \otimes \left(|0\rangle + e^{2\pi i \cdot 0.j_{m-2}j_{m-3}\cdots j_0}|1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + e^{2\pi i \cdot 0.j_0}|1\rangle\right)}{\sqrt{2^m}}
\end{aligned} \tag{4.2}$$

この変形により，回転ゲート $R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix}$ と Hadamard ゲート $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ を用いて，量子フーリエ変換の量子回路は図 4.1 のように効率的に実装可能であることがわかる．ただし，簡単のため出力における正規化のための $1/\sqrt{2}$ 倍を省略している．

図 4.1 の量子回路に $|j_{m-1}j_{m-2}\cdots j_0\rangle$ を入力したときの動作を確認する．まず，$|j_{m-1}\rangle$ に Hadamard ゲートを適用すると，量子状態は

$$\frac{|0\rangle + e^{2\pi i \cdot 0.j_{m-1}}|1\rangle}{\sqrt{2}} \otimes |j_{m-2}\cdots j_0\rangle$$

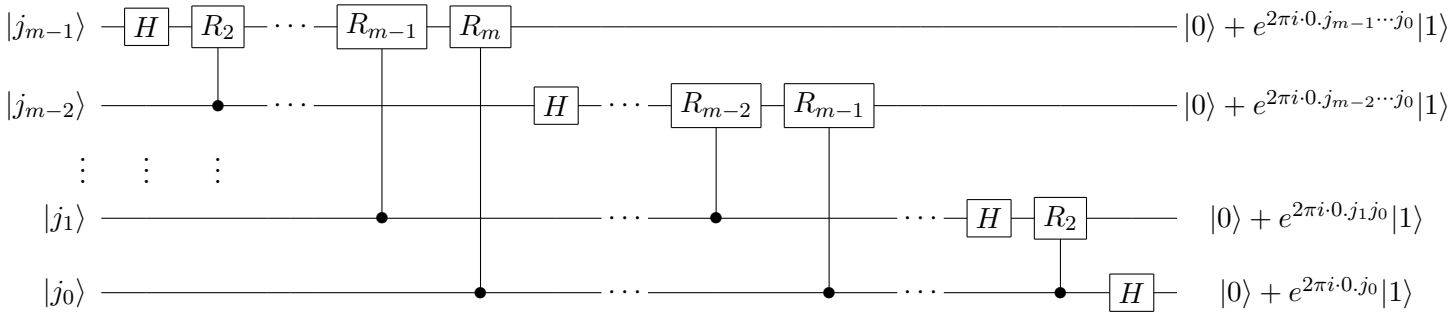

図 4.1: フーリエ変換の量子回路

となる．これは，式 (3.3) と $e^{2\pi i \cdot 0.1} = e^{\pi i} = -1$ から確認できる．そして，制御 R_2 ゲートを適用すると，量子状態は

$$\frac{|0\rangle + e^{2\pi i \cdot 0.j_{m-1}j_{m-2}}|1\rangle}{\sqrt{2}} \otimes |j_{m-2}\cdots j_0\rangle$$

となり，同様に制御 $R_3, \ldots, R_m$ ゲートを適用することで量子状態は

$$\frac{|0\rangle + e^{2\pi i \cdot 0.j_{m-1}j_{m-2}\cdots j_0}|1\rangle}{\sqrt{2}} \otimes |j_{m-2}\cdots j_0\rangle$$

となる．残りのビットについても同様に考えることで，最終的な量子状態が式 (4.2) のように書けることがわかる．

図 4.1 からわかるように，m ビットの量子フーリエ変換を行うためには m 量子ビットと $m(m+1)/2$ 個の量子ゲートで十分である．古典的に離散フーリエ変換を行うためには高速フーリエ変換のような現在知られている最も高速なアルゴリズムで $\Theta(m2^m)$ 個のゲート演算が必要であるので，量子フーリエ変換は指数的に効率化されていることがわかる．また，最終的な出力に最大 $m/2$ 個のスワップ操作をすることにより，量子フーリエ変換は

$$|j\rangle \to \frac{\left(|0\rangle + e^{2\pi i \cdot 0.j_{m-1}j_{m-2}\cdots j_0}|1\rangle\right) \otimes \left(|0\rangle + e^{2\pi i \cdot 0.j_{m-2}j_{m-3}\cdots j_0}|1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + e^{2\pi i \cdot 0.j_0}|1\rangle\right)}{\sqrt{2^m}} \quad (4.3)$$

となる操作と記述することができる．

4.3 位数計算アルゴリズム

第 4.1 節で述べたように，本節で Shor のアルゴリズムのための位数計算アルゴリズムについて説明する．第 4.3.1 小節で位相推定アルゴリズムを説明し，第 4.3.2 小節で素因数分解のための位数計算アルゴリズムについて説明する．

4.3.1 位相推定アルゴリズム

ユニタリ変換 U は，固有値 $e^{2\pi i\varphi}$ の固有ベクトル $|u\rangle$ を持ち，φ は未知であるとする．つまり，

$$U|u\rangle = e^{2\pi i\varphi}|u\rangle \tag{4.4}$$

を満たす．位相推定アルゴリズムは，φ を推定するアルゴリズムである．ここで，量子状態 $|u\rangle$ を用意することが可能で，非負整数 j に対して制御 U^{2^j} ゲートを効率的に演算可能であると仮定する．位相推定アルゴリズムは二つのレジスタからなり，第 1 レジスタには最初は m ビットの量子状態 $|0\cdots0\rangle$ を持つ．このとき，m の決め方については後述する．第 2 レジスタには最初は量子状態 $|u\rangle$ を持ち，それを表現するために必要なビット数を持つ．図 4.2 に位相推定アルゴリズムの量子回路を記す．

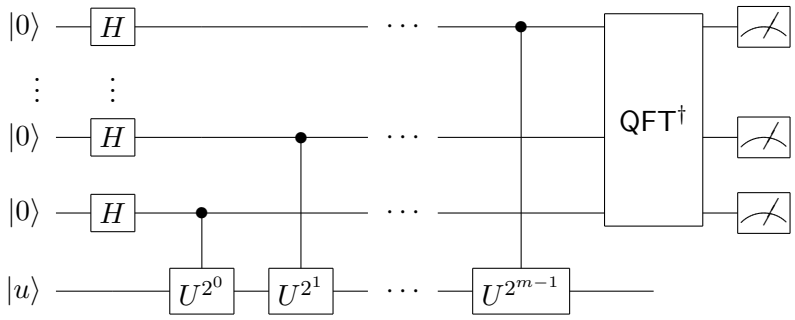

図 4.2: 位相推定アルゴリズムの量子回路

まず，$|0\cdots0\rangle|u\rangle$ を入力として第 1 レジスタに Hadamard ゲートを適用し，第 2 レジスタに $j = 0, 1, \ldots, m-1$ の制御 U^{2^j} ゲートを適用する．このとき，第 1 レジスタの量子状態は

$$\frac{\left(|0\rangle + e^{2\pi i\cdot 2^{m-1}\varphi}|1\rangle\right) \otimes \left(|0\rangle + e^{2\pi i\cdot 2^{m-2}\varphi}|1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + e^{2\pi i\cdot 2^{0}\varphi}|1\rangle\right)}{\sqrt{2^m}} \tag{4.5}$$

で，第 2 レジスタの量子状態は $|u\rangle$ のままであることを確認する．第 1 レジスタの最後の量子ビットは，$|0\rangle$ を入力として Hadamard 変換の後には量子状態は $(|0\rangle + |1\rangle)/\sqrt{2}$ となっている．そこに制御 U^{2^0} ゲートを適用すると，第 2 レジスタとの量子状態は式 (4.4) より

$$\begin{aligned}\frac{|0\rangle + |1\rangle}{\sqrt{2}}|u\rangle &\to \frac{|0\rangle}{\sqrt{2}}|u\rangle + \frac{|1\rangle}{\sqrt{2}}e^{2\pi i\cdot 2^0\varphi}|u\rangle \\ &= \frac{|0\rangle + e^{2\pi i\cdot 2^0\varphi}|1\rangle}{\sqrt{2}}|u\rangle\end{aligned}$$

となり，このとき第 1 レジスタの最初の $m-1$ ビットは変化しない．同様に，第 1 レジスタの下から j ビット目と第 2 レジスタは，

$$\begin{aligned}|0\rangle|u\rangle &\to \frac{|0\rangle + |1\rangle}{\sqrt{2}}|u\rangle \\ &\to \frac{|0\rangle}{\sqrt{2}}|u\rangle + \frac{|1\rangle}{\sqrt{2}}e^{2\pi i\cdot 2^{j-1}\varphi}|u\rangle \\ &= \frac{|0\rangle + e^{2\pi i\cdot 2^{j-1}\varphi}|1\rangle}{\sqrt{2}}|u\rangle\end{aligned}$$

と変化し第 1 レジスタの他のビットは変化しない．これらをまとめると，第 2 レジスタに $j = 0, 1, \ldots, m-1$ の制御 U^{2^j} ゲートを適用した後の第 1 レジスタの量子状態は式 (4.5) となることがわかる．図 4.2 より，制御 U^{2^j} ゲートを適用した後には量子逆フーリエ変換 $\mathsf{QFT}^\dagger$ を適用する．量子逆フーリエ変換は量子フーリエ変換の回路を逆にして各ゲートを共役転置にすることで得られるため，ゲート数は $\Theta(m^2)$ 個で十分である．量子逆フーリエ変換を適用した後には計算基底で第 1 レジスタを測定する．

これらの操作によって位相推定が可能であることを確かめるため，位相 φ が m ビットで $\varphi = 0.\varphi_1 \cdots \varphi_m$ と書けるとする．このとき，第 2 レジスタに $j = 0, 1, \ldots, m-1$ の制御 U^{2^j} ゲートを適用した後の第 1 レジスタの量子状態 (4.5) を

$$\begin{aligned}&\frac{\left(|0\rangle + e^{2\pi i\cdot 0.\varphi_m}|1\rangle\right) \otimes \left(|0\rangle + e^{2\pi i\cdot 0.\varphi_{m-1}\varphi_m}|1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + e^{2\pi i\cdot 0.\varphi_1\cdots\varphi_m}|1\rangle\right)}{\sqrt{2^m}} \\ &= \frac{1}{\sqrt{2^m}}\sum_{k=0}^{2^m-1} e^{2\pi i\varphi k}|k\rangle\end{aligned} \tag{4.6}$$

と書くことができる．量子逆フーリエ変換は，式 (4.3) の右辺を左辺にする操作であるので，式 (4.6)

の量子状態に量子逆フーリエ変換を適用した後の量子状態は

$$\frac{\left(|0\rangle + e^{2\pi i \cdot 0.\varphi_m}|1\rangle\right) \otimes \left(|0\rangle + e^{2\pi i \cdot 0.\varphi_{m-1}\varphi_m}|1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + e^{2\pi i \cdot 0.\varphi_1 \cdots \varphi_m}|1\rangle\right)}{\sqrt{2^m}} \to |\varphi_1 \cdots \varphi_m\rangle$$

と書くことができる．よって，この量子状態を測定することでユニタリ変換 U の位相 ϕ を測定することができる．

これまで述べた方法で，m ビットの φ を計算することができるが，ここから一般の場合には φ の近似値 $\tilde{\varphi}$ を高確率で得られることを確認する．$0 \leq b \leq 2^m - 1$ を満たす整数 b は，φ より小さい $b/2^m = 0.b_1 \cdots b_m$ が ϕ の最も良い m ビット近似値とする．つまり，$\delta \coloneqq \varphi - b/2^m$ は $0 \leq \delta \leq 2^{-m}$ を満たす．位相推定アルゴリズムによって観測される値は b に近く，φ の良い近似となることを確認する．式 (4.6) の量子状態に量子逆フーリエ変換を適用した後の量子状態は

$$\frac{1}{\sqrt{2^m}} \sum_{k=0}^{2^m-1} e^{2\pi i \varphi k}|k\rangle \to \frac{1}{2^m} \sum_{\ell=0}^{2^m-1} \sum_{k=0}^{2^m-1} e^{-2\pi i k \ell} e^{2\pi i \varphi k}|\ell\rangle$$

となる．α_ℓ を $|(b+\ell) \mod 2^m\rangle$ の係数すると，

$$\begin{aligned}
\alpha_\ell &\coloneqq \frac{1}{2^m} \sum_{k=0}^{2^m-1} \left(e^{2\pi i(\varphi - (b+\ell)/2^m)}\right)^k \\
&= \frac{1}{2^m} \left(\frac{1 - e^{2\pi i(2^m\varphi - (b+\ell))}}{1 - e^{2\pi i(\varphi - (b+\ell)/2^m)}}\right) \\
&= \frac{1}{2^m} \left(\frac{1 - e^{2\pi i(2^m\delta - \ell)}}{1 - e^{2\pi i(\delta - \ell/2^m)}}\right)
\end{aligned} \tag{4.7}$$

となる．最終的に観測される値を t としたとき，$|t-b| > e$ となる確率を考える．その確率は

$$\Pr[|t-b| > e] = \sum_{\ell=-2^{m-1}+1}^{-(e+1)} |\alpha_\ell|^2 + \sum_{\ell=e+1}^{2^{m-1}} |\alpha_\ell|^2 \tag{4.8}$$

となる．任意の θ に対して $|1 - e^{i\theta}| \leq 2$ となるので，式 (4.7) より

$$|\alpha_\ell| \leq \frac{2}{2^m |1 - e^{2\pi i(\delta - \ell/2^m)}|}$$

が成り立つ．$-\pi \leq \theta \leq \pi$ のとき常に $|1 - e^{i\theta}| \geq 2|\theta|/\pi$ が成り立つ．さらに，$0 \leq \delta \leq 2^{-m}$ より $-2^{m-1} + 1 \leq \ell \leq 2^{m-1}$ のとき $-\pi \leq 2\pi(\delta - \ell/2^m) \leq \pi$ となるので，

$$|\alpha_\ell| \leq \frac{1}{2^{m+1}(\delta - \ell/2^m)}$$

が成り立つ．これを式 (4.8) と組み合わせて，

$$\Pr[|t - b| > e] \leq \frac{1}{4}\left(\sum_{\ell=-2^{m-1}+1}^{-(e+1)} \frac{1}{(\ell - 2^m\delta)^2} + \sum_{\ell=e+1}^{2^{m-1}} \frac{1}{(\ell - 2^m\delta)^2}\right)$$

を得る．最後に，$0 \leq 2^m\delta \leq 1$ より，

$$\begin{aligned}
\Pr[|t - b| > e] &\leq \frac{1}{4}\left(\sum_{\ell=-2^{m-1}+1}^{-(e+1)} \frac{1}{\ell^2} + \sum_{\ell=e+1}^{2^{m-1}} \frac{1}{(\ell-1)^2}\right) \\
&\leq \frac{1}{2}\sum_{\ell=e}^{2^{m-1}-1} \frac{1}{\ell^2} \\
&\leq \frac{1}{2}\int_{e-1}^{2^{m-1}-1} \frac{1}{\ell^2} d\ell \\
&= \frac{1}{2(e-1)}
\end{aligned}$$

を得る．つまり，φ を 2^{-n} の精度で近似するとき，$e = 2^{m-n} - 1$ とし，$m = n + p$ とすると，上式よりその精度で近似値が得られる確率は $1 - 1/(2(2^p - 2))$ となる．つまり，φ の n ビット近似を確率 $1 - \varepsilon$ 以上で得るためには，

$$m = n + \left\lceil \log\left(2 + \frac{1}{2\varepsilon}\right)\right\rceil$$

とすれば良い．

4.3.2 素因数分解のための位数計算アルゴリズム

これより，第 4.1 節で述べた位数計算アルゴリズムは，第 4.3.1 小節の位相推定アルゴリズムと同様にして効率的な量子アルゴリズムを得られることを説明する．ここでは，第 4.1 節における素因数

分解を考え，$a^r = 1 \mod N$ を満たす最小の正整数 r を求める場合を考える．そのために，以下のユニタリ変換

$$U_a|x\rangle = |ax \mod N\rangle$$

を考える．$\gcd(a, N) \neq 1$ のとき，逆変換 $a^{-1} \mod N$ が存在するため U_a が確かにユニタリ変換である．ただし，x は N と同様 n ビットであるとし，$N \leq x \leq 2^n - 1$ のときには $ax \mod N = x$ とする．つまり，ユニタリ変換 U_a が非自明な操作となるのは $0 \leq x \leq N - 1$ のときのみである．ここで，$0 \leq s \leq r - 1$ を満たす整数 s に対して定義される量子状態

$$|u_s\rangle \coloneqq \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi isk/r} |a^k \mod N\rangle$$

は，$a^r = 1 \mod N$ が成り立つことに注意すると

$$\begin{aligned}
U_a|u_s\rangle &= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi isk/r} |a^{k+1} \mod N\rangle \\
&= \frac{1}{\sqrt{r}} \sum_{\ell=1}^{r} e^{-2\pi is(\ell-1)/r} |a^{\ell} \mod N\rangle \\
&= e^{2\pi is/r} \cdot \frac{1}{\sqrt{r}} \sum_{\ell=1}^{r} e^{-2\pi is\ell/r} |a^{\ell} \mod N\rangle \\
&= e^{2\pi is/r} \cdot \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi isk/r} |a^{k} \mod N\rangle \\
&= e^{2\pi is/r} |u_s\rangle
\end{aligned}$$

が成り立ち，量子状態 $|u_s\rangle$ はユニタリ変換 U_a の固有状態であることがわかる．この変形は $e^{-2\pi isr/r} = 1 = e^{-2\pi is0/r}$ と $a^r \mod N = 1 = a^0 \mod N$ によって得られる．

上記の議論より，量子状態 $|u_s\rangle$ を用意することが可能で，非負整数 j に対して制御 $U_a^{2^j}$ ゲートを効率的に演算可能であるならば位相推定アルゴリズムによって s/r を計算可能であることがわかる．ところが，量子状態 $|u_s\rangle$ を用意するためには求めたい位数 r を知っている必要があるため，一つ目の制約は自明には成り立たない．ただし，k が r の倍数ではないときに

$$\sum_{s=0}^{r-1} e^{-2\pi isk/r} = 0$$

が成り立つことより，

$$\begin{aligned}\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}|u_s\rangle &= \frac{1}{r}\sum_{s=0}^{r-1}\sum_{k=0}^{r-1}e^{-2\pi isk/r}|a^k \mod N\rangle \\ &= \frac{1}{r}\sum_{s=0}^{r-1}|1\rangle \\ &= |1\rangle\end{aligned}$$

が成り立ち，量子状態 $|1\rangle$ を用意することが可能である．そのため，第 1 レジスタの量子ビット数を $m = 2n + 1 + \lceil \log\left(2 + \frac{2}{2\varepsilon}\right) \rceil$ とし，第 2 レジスタの量子状態を $|1\rangle$ として位相推定アルゴリズムを適用すると，各 $s = 0, 1, \ldots, r-1$ における $\varphi \approx s/r$ の $2n+1$ ビット近似値を確率 $(1-\varepsilon)/r$ で得ることができる．詳細は省略するが，s/r が得られれば連分数展開により r を計算することができる．位数計算アルゴリズムの量子回路を図 4.3 に記す．

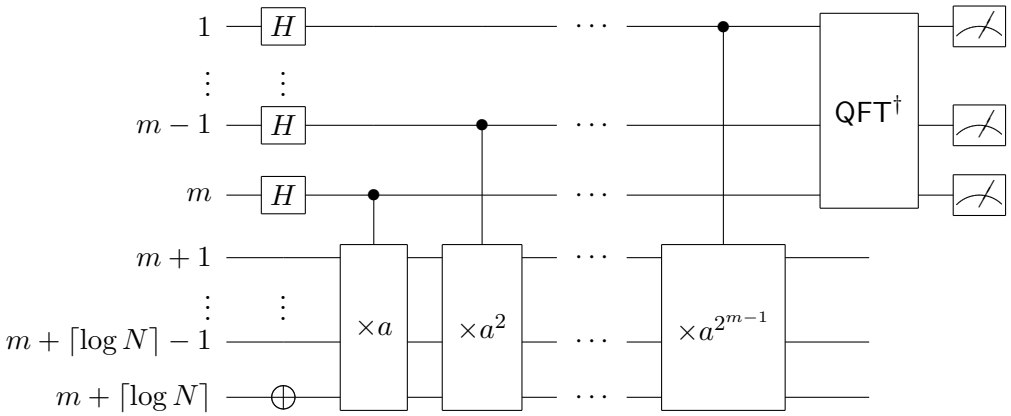

図 4.3: 素因数分解のための位数計算アルゴリズムの量子回路．$\times a, \times a^2, \ldots, a^{2^{m-1}}$ はそれぞれ $\times a \mod N, \times a^2 \mod N, \ldots, a^{2^{m-1}} \mod N$ を計算する量子ゲートである．

次に，非負整数 j に対して制御 $U_a^{2^j}$ ゲートを効率的に演算可能であるであることを確認する．その前に，この操作は第 1 レジスタが 2 進数で表された整数 x であるときに，第 2 レジスタに冪乗剰余演算 $a^x \mod N$ の計算結果を書き込むものである．つまり，第 1 レジスタの m ビットが $x_{m-1}, x_{m-2}, \ldots, x_0$ であるとき，それらをそれぞれ制御ビットとして $a^{2^{m-1}} \mod N, a^{2^{m-2}} \mod N, \ldots, a \mod N$ 倍す

ることは

$$a^{x_0} \cdot (a^2)^{x_1} \cdot (a^{2^2})^{x_2} \cdot \cdots + (2^{m-1})^{x_{m-1}} \mod N$$

を計算することであり，m ビット整数 x を

$$x = x_0 + 2x_1 + 2^2 x_2 + \cdots + 2^{m-1} x_{m-1}$$

と書くとき，第 2 レジスタの計算結果は

$$\begin{aligned} & a^{x_0} \cdot (a^2)^{x_1} \cdot (a^{2^2})^{x_2} \cdot \cdots + (2^{m-1})^{x_{m-1}} \mod N \\ = \; & a^{x_0+2x_1+2^2x_2+\cdots+2^{m-1}x_{m-1}} \mod N \\ = \; & a^x \mod N \end{aligned}$$

となっている．また，前述の通り量子逆フーリエ変換は $\Theta(n^2)$ 個の量子ゲートで十分だが，冪乗剰余演算には $O(n^3)$ 個の量子ゲートが必要であり，冪乗剰余演算が素因数分解の量子アルゴリズムで最も計算が重いステップである．

4.4 周期計算アルゴリズム

この節で，まず周期計算アルゴリズムの説明を行う．その後，第 4.4.1 小節で素体上の離散対数アルゴリズムを，第 4.4.2 で楕円曲線上の離散対数アルゴリズムを説明する．

1 ビット出力する関数 f は，$0 < r < 2^n$ を満たす周期 r を持つとする．つまり，全ての整数 x に対して

$$f(x + r) = f(x)$$

が成り立つ．$U|x\rangle|y\rangle \to |x\rangle|y \oplus x\rangle$ を計算するユニタリ演算子 U が与えられたとき，この周期 r を求めたい．そのために初期状態を $|0\rangle|0\rangle$ とする二つのレジスタを用意する．ただし，第 1 レジスタは $m = O(n + \log(1/\varepsilon))$ ビットとする．このとき，前節の位相推定アルゴリズムとほぼ同じ流れで周期 r を計算することができる．このアルゴリズムの成功確率は $O(1)$ で，U を一度と他に $O(n^2)$ 回の演算を要する．

初期状態を $|0\rangle|0\rangle$ とし，第 1 レジスタに Hadamard 変換をかけることで，量子状態を

$$|0\rangle|0\rangle \to \frac{1}{\sqrt{2^m}} \sum_{x=0}^{2^m-1} |x\rangle|0\rangle$$

とする．次に，ユニタリ変換 U を適用することで量子状態を

$$\frac{1}{\sqrt{2^m}} \sum_{x=0}^{2^m-1} |x\rangle|f(x)\rangle \tag{4.9}$$

とする．ここで，量子状態 $|f(x)\rangle$ のフーリエ変換となる量子状態

$$|\hat{f}(\ell)\rangle \coloneqq \frac{1}{\sqrt{r}} \sum_{x=0}^{r-1} e^{-2\pi i \ell x/r} |f(x)\rangle$$

を考えると，

$$|f(x)\rangle = \frac{1}{\sqrt{r}} \sum_{\ell=0}^{r-1} e^{2\pi i \ell x/r} |\hat{f}(\ell)\rangle$$

と書くことができる．これにより，式 (4.9) の量子状態を

$$\approx \frac{1}{\sqrt{r2^m}} \sum_{\ell=0}^{r-1} \sum_{x=0}^{2^m-1} e^{2\pi i \ell x/r} |x\rangle|\hat{f}(\ell)\rangle$$

と近似することができる．第 1 レジスタに量子フーリエ逆変換を適用すると，量子状態は

$$\frac{1}{\sqrt{r}} \sum_{\ell=0}^{r-1} |\widetilde{\ell/r}\rangle|\hat{f}(\ell)\rangle$$

となり，第 1 レジスタを測定することでランダムな ℓ に対する ℓ/r の近似値 $\widetilde{\ell/r}$ を得ることができ，そこから連分数展開により r の値を計算することができる．

4.4.1 素体上の離散対数アルゴリズム

周期計算アルゴリズムから離散対数アルゴリズムを構成するのは自明ではない．第 4.1 節で述べたとおり，離散対数では二次元の周期を求める必要がある．だが，離散対数問題も周期計算アルゴリズムのように，確率 $O(1)$ で $U|x_1\rangle|x_2\rangle|y\rangle \to |x_1\rangle|x_2\rangle|y \oplus f_{g,h}(x_1,x_2)\rangle = |x_1\rangle|x_2\rangle|y \oplus h^{x_1} \cdot g^{x_2}\rangle$ なるユニタリ変換 U を一度と $O(\lceil \log r \rceil^2)$ 回の演算で解くことができる．ただし，r は $g^r \mod N = 1$ を満たす最小の自然数で，r の値はわかっているとする．

初期状態を $|0\rangle|0\rangle|0\rangle$ とする．ただし，最初の二つのレジスタは $m = \lceil \log r \rceil + \log(1/\varepsilon)$ ビットで，第 3 レジスタは $\lceil \log N \rceil$ ビットである．最初の二つのレジスタに Hdamard 変換をかけることで，量子状態を

$$\frac{1}{2^m} \sum_{x_1=0}^{2^m-1} \sum_{x_2=0}^{2^m-1} |x_1\rangle|x_2\rangle|0\rangle$$

とする．次に，ユニタリ変換 U を適用することで量子状態を

$$\frac{1}{2^m} \sum_{x_1=0}^{2^m-1} \sum_{x_2=0}^{2^m-1} |x_1\rangle|x_2\rangle|f_{g,h}(x_1, x_2)\rangle \tag{4.10}$$

とする．ここで，量子状態 $|f(x_1, x_2)\rangle$ のフーリエ変換となる量子状態

$$\begin{aligned} |\hat{f}_{g,h}(\ell_1, \ell_2)\rangle &= \sum_{x_1=0}^{r-1} \sum_{x_2=0}^{r-1} e^{-2\pi i(\ell_1 x_1 + \ell_2 x_2)/r} |f(x_1, x_2)\rangle \\ &= \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} e^{-2\pi i \ell_2 j/r} |f_{g,h}(0, j)\rangle \end{aligned}$$

を考える．ただし，ℓ_1 と ℓ_2 は

$$\sum_{k=0}^{r-1} e^{2\pi i k(\ell_1/s - \ell_2)/r} = r$$

を満たす．つまり，$\ell_1/s - \ell_2$ は r の整数倍である．すると，式 (4.10) の量子状態を

$$\begin{aligned} &\approx \frac{1}{2^m\sqrt{r}} \sum_{\ell_2=0}^{r-1} \sum_{x_1=0}^{2^m-1} \sum_{x_2=0}^{2^m-1} e^{2\pi i(s\ell_2 x_1 + \ell_2 x_2)/r} |x_1\rangle|x_2\rangle|\hat{f}_{g,h}(s\ell_2, \ell_2)\rangle \\ &= \frac{1}{2^m\sqrt{r}} \sum_{\ell_2=0}^{r-1} \left(\sum_{x_1=0}^{2^m-1} e^{2\pi i s\ell_2 x_1/r} |x_1\rangle \right) \left(\sum_{x_2=0}^{2^m-1} e^{2\pi i \ell_2 x_2/r} |x_2\rangle \right) |\hat{f}_{g,h}(s\ell_2, \ell_2)\rangle \end{aligned}$$

と近似することができる．第 1 レジスタと第 2 レジスタそれぞれに量子フーリエ逆変換を適用すると，量子状態は

$$\frac{1}{\sqrt{r}} \sum_{\ell_2=0}^{r-1} |\widetilde{s\ell_2/r}\rangle|\widetilde{\ell_2/r}\rangle|\hat{f}_{g,h}(s\ell_2, \ell_2)\rangle$$

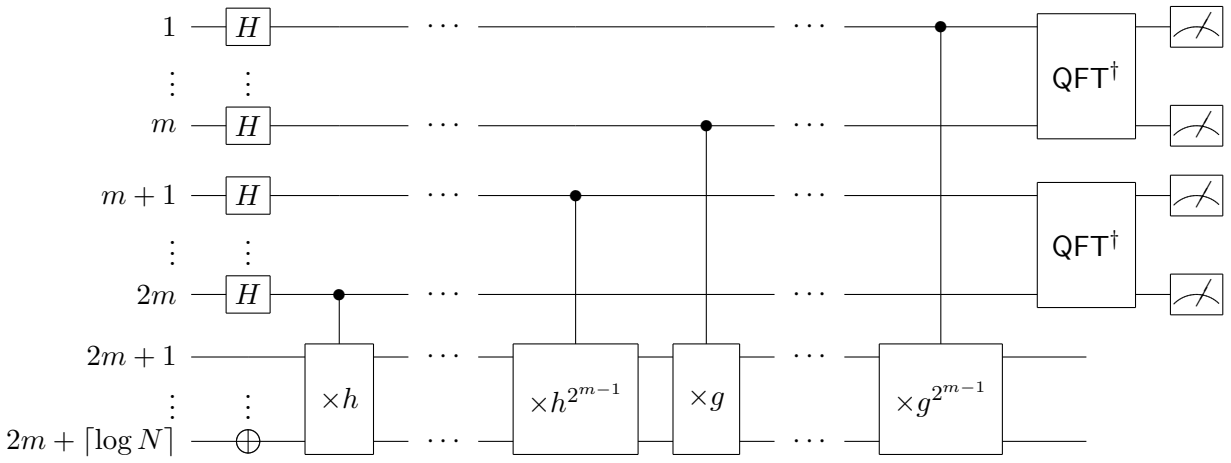

図 4.4: 離散対数問題のための周期計算アルゴリズムの量子回路．$\times h, \ldots, h^{2^{m-1}}$ と $\times g, \ldots, g^{2^{m-1}}$ はそれぞれ $\times h \mod N, \ldots, h^{2^{m-1}} \mod N$ と $\times g \mod N, \ldots, g^{2^{m-1}} \mod N$ を計算する量子ゲートである．

となり，第 1 レジスタと第 2 レジスタを測定することでランダムな ℓ_2 に対する近似値 $\widetilde{s\ell_2/r}$ と $\widetilde{\ell_2/r}$ を得ることができ，そこから連分数展開により s の値を計算することができる．

図 4.4 は，離散対数問題を解くための量子回路である．ただし，第 3 レジスタには一度のユニタリ変換 $f_{g,h}(x_1, x_2)$ の結果ではなく，まず最後のビットを反転させて量子状態を $|1\rangle$ とした後に，素因数分解の場合と同じように第 1 レジスタと第 2 レジスタを各ビットごとに $h, h^2, \ldots, h^{2^{m-1}}$ と $g, g^2, \ldots, g^{2^{m-1}}$ の制御乗算を行うことで $h^{x_1} \cdot g^{x_2}$ を計算している．素因数分解アルゴリズムの場合と同様，$h^{x_1} \cdot g^{x_2}$ の計算には $O(n^3)$ 個の量子ゲートが必要であり，素体上の素因数分解の量子アルゴリズムで最も計算が重いステップである．

4.4.2 楕円曲線上の離散対数アルゴリズム

楕円曲線上の離散対数問題（elliptic curve discrete logarithm problem, ECDLP）は，楕円曲線上で定義される離散対数問題である．ECDLP の定義を [RNSL17] の Section 2 に倣い説明する．$p > 3$ を満たす素数 p に対して，$\mathbb{F}_p$ 上で定義される Weierstrass 標準形は $E : y^2 = x^3 + ax + b$ の全ての解

(x, y) の集合と無限遠点 $\mathcal{O}$ であり，

$$E(\mathbb{F}_p) \coloneqq \{(x, y) \in \mathbb{F}_p \times \mathbb{F}_p | y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$$

と書くことができる．ただし，$a, b \in \mathbb{F}_p$ は曲線を定義する定数である．詳細は省略するがこの曲線は，加法演算によって可換群となる．ECDLP は，位数 r の元 $P \in E(\mathbb{F}_p)$ と $Q \in \langle P \rangle$ を入力として $Q = kP$ を満たす $k \in \{1, 2, \ldots, r\}$ を求める問題である．ECDLP を解くための量子アルゴリズムは，素体上の離散対数問題アルゴリズムと同様である．初期状態を $|0\rangle|0\rangle|0\rangle$ とする．ただし，最初の二つのレジスタは $m = \lceil \log r \rceil + \log(1/\varepsilon)$ ビットで，第 3 レジスタは $\lceil \log p \rceil$ ビットである．最初の二つのレジスタに Hdamard 変換をかけることで，量子状態を

$$\frac{1}{2^m} \sum_{x_1=0}^{2^m-1} \sum_{x_2=0}^{2^m-1} |x_1\rangle|x_2\rangle|0\rangle$$

とする．次に，

$$\frac{1}{2^m} \sum_{x_1=0}^{2^m-1} \sum_{x_2=0}^{2^m-1} |x_1\rangle|x_2\rangle|x_1 P + x_2 Q\rangle$$

を計算する．さらに，第 1 レジスタと第 2 レジスタそれぞれに量子フーリエ逆変換を適用して測定することで ECDLP を解くことができる．ECDLP を解くための量子回路を図 4.5 に示す．

図 4.5 は，ECDLP を解くための量子回路である．ただし，第 3 レジスタでは一度のユニタリ変換ではなく，$P, 2P, \ldots, 2^{m-1}P$ と $Q, 2Q, \ldots, 2^{m-1}Q$ の制御加算を行うことで $x_1 P + x_2 Q$ を計算している．これまでの場合と同様，$x_1 P + x_2 Q$ の計算には $O(n^3)$ 個の量子ゲートが必要であり，ECDLP の量子アルゴリズムで最も計算が重いステップである．

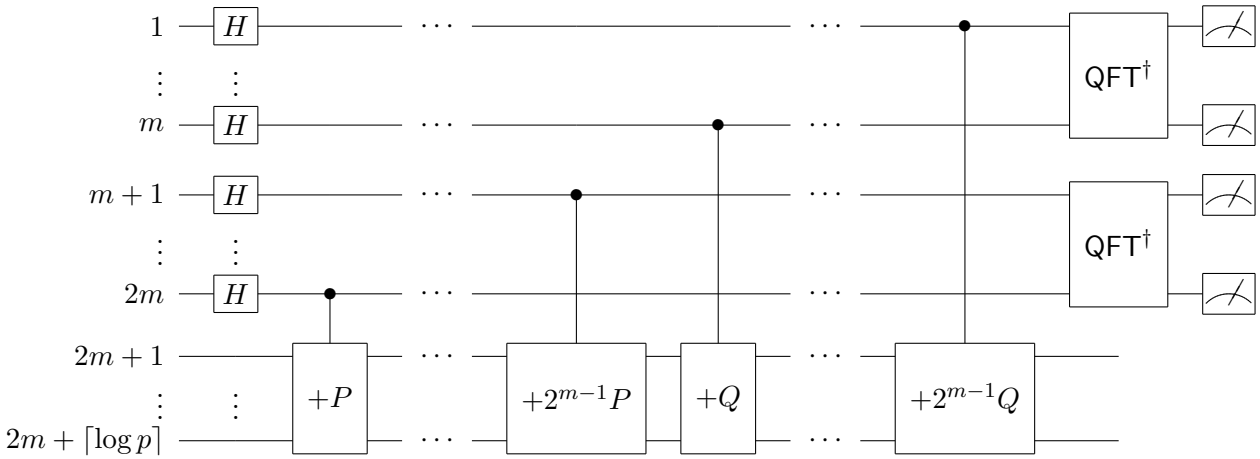

図 4.5: ECDLP のための周期計算アルゴリズムの量子回路．$+P, \ldots, +2^{m-1}P$ と $+Q, \ldots, +2^{m-1}Q$ はそれぞれ楕円曲線 $E(\mathbb{F}_p)$ 上で $+P, \ldots, +2^{m-1}P$ と $+Q, \ldots, +2^{m-1}Q$ を計算する量子ゲートである．

第5章　実装結果

この章で，これまで行われた Shor の素因数分解アルゴリズムと素体上の離散対数アルゴリズムの実装実験についてまとめる．具体的には，$N = 15, 21$ の素因数分解実験 [ASK19, DLQ+20, LBC+12, LBYP07, MLLL+12, LWL+07, MNM+16, PMO09, SSV13, VSB+01] と $2^x = 1 \mod 3$ の離散対数問題実験 [AST+20] の現在知られている結果をまとめる．ほとんどの実験は $N = 15$ の素因数分解であるため，実験で用いる量子回路の理解が容易になるように，第 5.1 節で $N = 15$ のときの素因数分解について簡単にまとめる．第 5.2 節では，これらの実験を成功させるために行われた効率化手法をまとめる．そして第 5.3 節で，既存の実験結果についてまとめる．

5.1　15 の素因数分解

第 4.3.2 小節で Shor の素因数分解における量子パートを一般の N の場合で説明したが，この節ではより理解が深まるように $N = 15$ の場合を説明する．これによって，素因数分解アルゴリズムの理解が深まるのみならず，$N = 15$ は特別に実装しやすい合成数であることが確認できる．

$N = 15$ を素因数分解するために，初期状態

$$|0 \cdots 0\rangle |0 \cdots 0\rangle$$

を用意する．ここで，第 1 レジスタを m ビット，第 2 レジスタを $\lceil \log 15 \rceil = 4$ ビットとする．第 1 レジスタに Hadamard ゲート，最下位ビットに NOT ゲートを適用することで量子状態が

$$\frac{1}{\sqrt{2^m}} \sum_{i=0}^{2^m-1} |i\rangle |0 \cdots 01\rangle$$

となる．さらに，冪乗剰余演算ゲートを適用することで

$$\frac{1}{\sqrt{2^m}} \sum_{i=0}^{2^m-1} |i\rangle |a^i \mod 15\rangle$$

を得る．第 4.3.2 小節で述べた通り，この冪乗剰余演算が素因数分解アルゴリズムで最も計算が重いステップであるため，15 の場合にどれほどの演算が必要なのかを確認する．

第 4.1 小節のステップ 1 において，a として 15 とは互いに素な整数 $\{2, 4, 7, 8, 11, 13, 14\}$ のいずれかが選ばれる．例として $a \in \{2, 4, 7, 8, 11, 13\}$ のときに $a \mod 15$ 倍を計算する量子回路を図 5.1 に示す．第 4.3.2 節で述べたように，冪乗演算を行うために一般には $a, a^2, \ldots, a^{2^{m-1}} \mod 15$ 倍の計算をする必要がある．ただし，これらの値は一般の合成数 N の素因数分解の場合においても古典的に $a^j \mod N$ の値を事前計算したうえで各量子ゲートを構成することができる．$N = 15$ の場合にこれらがどのような値になるのかを確認する．まず，全ての $a \in \{2, 4, 7, 8, 11, 13, 14\}$ に対して，$a^4 \mod 15 = 1$ が成り立つことがわかる．そのため，$a^4, a^8, \ldots, a^{2^n} \mod 15$ 倍を計算する量子ゲートは常に省略することができる．よって，第 4.3.2 小節で確認したように，n ビット合成数の素因数分解の量子アルゴリズムを高確率で実行するためには，$m > 2n$ を満たす m 個の制御乗法演算が必要であったが，$N = 15$ の場合には $a \mod 15$ 倍と $a^2 \mod 15$ 倍のたかだか 2 個で十分である．さらに，$a \in \{4, 11, 14\}$ のときには $a^2 = 1 \mod 15$ が成り立つため，このときの位数は 2 で，$a^2 \mod 15$ 倍を計算するゲートも省略可能で，必要なのは $a \mod 15$ 倍を計算する一つの量子ゲートのみである．その他の $a \in \{2, 7, 8, 13\}$ の位数は $r = 4$ であり，さらにこのとき全ての a に対して $a^2 = 4 \mod 15$ が成り立つため $a^2 \mod 15$ 倍を計算するゲートは $4 \mod 15$ 倍を計算するゲートで置き換え可能である．そのため，位数が 2 のときと 4 のときで 15 を素因数分解をする量子回路はそれぞれ簡略化することができるので，それぞれ第 5.1.1 小節と第 5.1.2 小節でそれぞれ説明する．また，$a^4 \mod 15 = 1$ が成り立つことで大幅に演算回数を削減できたため，本来は $a \mod 15$ 倍と $4 \mod 15$ 倍の剰余乗算をしなければならなかったところが，第 5.3 節で説明する多くの実験では a 倍と 4 倍の乗算にできているのも量子回路を効率化できた一因である．言い換えると，図 5.1 で示したような一般的な量子ゲートを使っていない実験がほとんどである．

5.1.1 位数が $r = 2$ のときの素因数分解

$N = 15, r = 2$ の素因数分解について詳しく説明する．このとき，$a \in \{4, 11, 14\}$ である．この場合の量子回路を図 5.2 に記す．前述の通り，この回路では $\times a^2 \mod 15, \ldots, \times a^{2^{m-1}} \mod 15$ の量子ゲートは省略されている．ここでは，[VSB+01] の実験に倣い $a = 11$ の場合を説明する．

$\times 11 \mod 15$ のユニタリ変換の固有ベクトルは

$$|u_0\rangle \coloneqq \frac{1}{\sqrt{2}} \sum_{k=0}^{1} |11^k \mod 15\rangle = \frac{|1\rangle + |11\rangle}{\sqrt{2}}$$

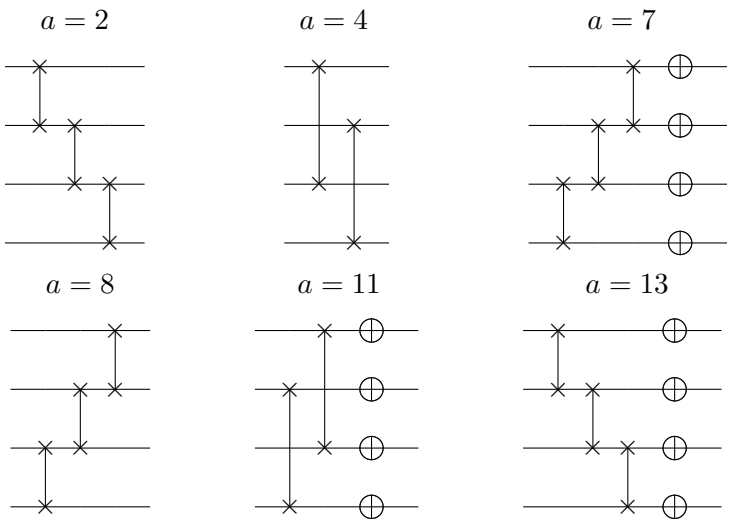

図 5.1: $a \in \{2, 4, 7, 8, 11, 13\}$ のときに a 倍を計算する回路

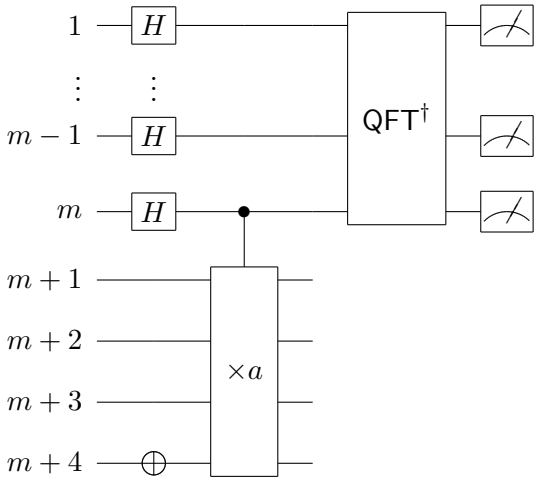

図 5.2: a の位数が 2 のときの 15 を素因数分解する量子回路

$$|u_1\rangle \coloneqq \frac{1}{\sqrt{2}}\sum_{k=0}^{1} e^{-\pi i k}|11^k \mod 15\rangle = \frac{|1\rangle - |11\rangle}{\sqrt{2}}$$

の二つであり，

$$\begin{aligned} U_{11}|u_0\rangle &= \frac{|11\rangle + |1\rangle}{\sqrt{2}} = |u_0\rangle \\ U_{11}|u_1\rangle &= \frac{|11\rangle - |1\rangle}{\sqrt{2}} = e^{\pi i}|u_1\rangle \end{aligned} \tag{5.1}$$

より，それぞれの固有値は 1 と $e^{\pi i}$ である．第 4.3.2 小節で述べたように，

$$\frac{|u_0\rangle + |u_1\rangle}{\sqrt{2}} = |1\rangle$$

が成り立つことが確認できる．初期状態を $|0\cdots0\rangle|0\rangle$ とし，第 1 レジスタに Hadamard ゲート，第 2 レジスタの最後のビットに NOT ゲートを適用した後の量子状態は

$$\begin{aligned} &\frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \cdots \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes |1\rangle \\ =\ &\frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \cdots \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes (|u_0\rangle + |u_1\rangle) \end{aligned}$$

となる．さらに，第 1 レジスタの最後のビットを制御ビットとする $\times 11 \mod 15$ ゲートを適用することで，量子状態は

$$\begin{aligned} &\frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \cdots \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes |u_0\rangle \\ +\ &\frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \cdots \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + e^{\pi i}|1\rangle}{\sqrt{2}} \otimes |u_1\rangle \\ =\ &\frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \cdots \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes |u_0\rangle \\ +\ &\frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \cdots \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + e^{2\pi i \cdot 0.1}|1\rangle}{\sqrt{2}} \otimes |u_1\rangle \end{aligned}$$

となる．さらに，第 1 レジスタに量子逆フーリエ変換をかけることで，

$$\frac{|0\cdots00\rangle}{\sqrt{2^m}} \otimes |u_0\rangle + \frac{|0\cdots01\rangle}{\sqrt{2^m}} \otimes |u_1\rangle$$

を得る．このとき，第 1 レジスタを測定することで $|0\cdots00\rangle$ または $|0\cdots01\rangle$ を得る．これらはそれぞれスワップの後に整数で書くと $|0\rangle$ または $|2^{m-1}\rangle$ に対応し，周期は 2^{m-1} である．つまり，位数は $r = 2^m/2^{m-1} = 2$ となる．よって，$\gcd(11^{2/2} \pm 1, 15)$ を計算することで 15 の素因数 3 と 5 を得る．

5.1.2 位数が $r = 4$ のときの素因数分解

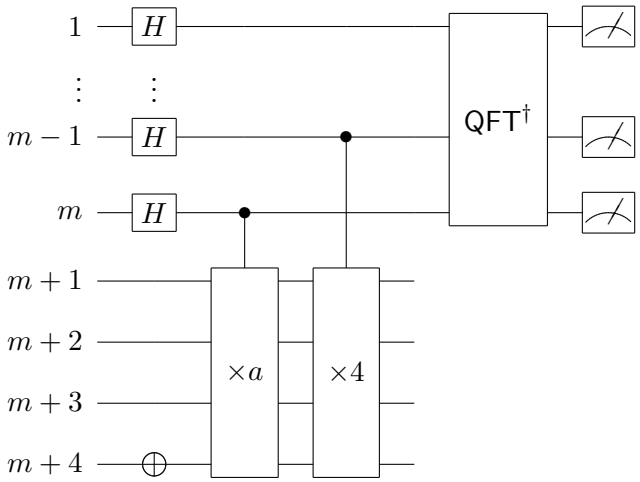

図 5.3: a の位数が 4 のときの 15 を素因数分解する量子回路

同様に，$N = 15, r = 4$ の素因数分解について詳しく説明する．このとき，$a \in \{2, 7, 8, 13\}$ である．この場合の量子回路を図 5.3 に記す．前述の通り，この回路では $\times a^4 \mod 15, \ldots, \times a^{2^m} \mod 15$ の量子ゲートは省略し，$\times a^2 \mod 15$ の量子ゲートを $\times 4 \mod 15$ で置き換えている．ここでも，[VSB+01] の実験に倣い $a = 7$ の場合を説明する．

$\times 7 \mod 15$ のユニタリ変換の固有ベクトルは

$$|u_0\rangle := \frac{1}{\sqrt{4}} \sum_{k=0}^{3} |7^k \mod 15\rangle = \frac{|1\rangle + |7\rangle + |4\rangle + |13\rangle}{\sqrt{4}}$$

$$|u_1\rangle := \frac{1}{\sqrt{4}} \sum_{k=0}^{3} e^{-\pi i k/2} |7^k \mod 15\rangle = \frac{|1\rangle - i|7\rangle - |4\rangle + i|13\rangle}{\sqrt{4}}$$

$$|u_2\rangle := \frac{1}{\sqrt{4}} \sum_{k=0}^{3} e^{-\pi i k} |7^k \mod 15\rangle = \frac{|1\rangle - |7\rangle + |4\rangle - |13\rangle}{\sqrt{4}}$$

$$|u_3\rangle := \frac{1}{\sqrt{4}} \sum_{k=0}^{3} e^{-3\pi i k/2} |7^k \mod 15\rangle = \frac{|1\rangle + i|7\rangle - |4\rangle - i|13\rangle}{\sqrt{4}}$$

の 4 つであり，

$$
\begin{aligned}
U_7|u_0\rangle &= \frac{|7\rangle + |4\rangle + |13\rangle + |1\rangle}{\sqrt{4}} = |u_0\rangle \\
U_7|u_1\rangle &= \frac{|7\rangle - i|4\rangle - |13\rangle + i|1\rangle}{\sqrt{4}} = e^{\pi i/2}|u_1\rangle \\
U_7|u_2\rangle &= \frac{|7\rangle - |4\rangle + |13\rangle - |1\rangle}{\sqrt{4}} = e^{\pi i}|u_1\rangle \\
U_7|u_3\rangle &= \frac{|7\rangle + i|4\rangle - |13\rangle - i|1\rangle}{\sqrt{4}} = e^{3\pi i/2}|u_1\rangle
\end{aligned}
$$

より，それぞれの固有値は $1, e^{\pi i/2}, e^{\pi i}, e^{3\pi i/2}$ である．第 4.3.2 節で述べたように，

$$
\frac{|u_0\rangle + |u_1\rangle + |u_2\rangle + |u_3\rangle}{\sqrt{4}} = |1\rangle
$$

が成り立つことが確認できる．初期状態を $|0\cdots0\rangle|0\rangle$ とし，第 1 レジスタに Hadamard ゲート，第 2 レジスタの最後のビットに NOT ゲートを適用した後の量子状態は

$$
\begin{aligned}
&\frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \cdots \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes |1\rangle \\
=\ &\frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \cdots \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes (|u_0\rangle + |u_1\rangle + |u_2\rangle + |u_3\rangle)
\end{aligned}
$$

となる．さらに，第 1 レジスタの最後の二つのビットを制御ビットとする $\times 7 \mod 15, \times 4 \mod 15$ ゲートを適用することで，量子状態は

$$
\begin{aligned}
&\frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \cdots \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes |u_0\rangle \\
+\ &\frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \cdots \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + e^{\pi i}|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + e^{\pi i/2}|1\rangle}{\sqrt{2}} \otimes |u_1\rangle \\
+\ &\frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \cdots \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + e^{\pi i}|1\rangle}{\sqrt{2}} \otimes |u_2\rangle \\
+\ &\frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \cdots \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + e^{\pi i}|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + e^{3\pi i/2}|1\rangle}{\sqrt{2}} \otimes |u_3\rangle \\
=\ &\frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \cdots \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes |u_0\rangle
\end{aligned}
$$

$$
\begin{aligned}
&+ \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \cdots \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + e^{2\pi i \cdot 0.1}|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + e^{2\pi i \cdot 0.01}|1\rangle}{\sqrt{2}} \otimes |u_1\rangle \\
&+ \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \cdots \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + e^{2\pi i \cdot 0.1}|1\rangle}{\sqrt{2}} \otimes |u_2\rangle \\
&+ \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \cdots \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + e^{2\pi i \cdot 0.1}|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + e^{2\pi i \cdot 0.11}|1\rangle}{\sqrt{2}} \otimes |u_3\rangle
\end{aligned}
$$

となる．さらに，第 1 レジスタに量子逆フーリエ変換をかけることで，

$$
\frac{|0\cdots000\rangle}{\sqrt{2^m}} \otimes |u_0\rangle + \frac{|0\cdots001\rangle}{\sqrt{2^m}} \otimes |u_1\rangle + \frac{|0\cdots010\rangle}{\sqrt{2^m}} \otimes |u_2\rangle + \frac{|0\cdots011\rangle}{\sqrt{2^m}} \otimes |u_3\rangle
$$

を得る．このとき，第 1 レジスタを測定することで $|0\cdots00\rangle, |0\cdots001\rangle, |0\cdots010\rangle$ または $|0\cdots011\rangle$ を得る．これらはそれぞれスワップの後に整数で書くと $|0\rangle, |2^{m-1}\rangle, |2^{m-2}\rangle$ または $|2^{m-1} + 2^{m-2}\rangle$ に対応し，周期は 2^{m-2} である．つまり，位数は $r = 2^m/2^{m-2} = 4$ となる．よって，$\gcd(7^{4/2} \pm 1, 15)$ を計算することで 15 の素因数 3 と 5 を得る．

5.2 実験の効率化手法

第 5.1 で確認したように，$N = 15$ という特別な合成数を考えたときには量子回路を簡略化できる．さらに，第 5.3 節で紹介するこれまでの実験ではいくつかの共通する効率化手法が用いられてきたので，それについて説明する．

5.2.1 第 2 レジスタの圧縮

第 5.1 節で見たように，第 1 レジスタで表される整数を x としたとき，第 2 レジスタには $a^x \mod 15$ の値を書き込むために $\lceil \log 15 \rceil = 4$ ビットが必要となる．つまり一般には，N を素因数分解するために第 2 レジスタには $\lceil \log_2 N \rceil$ ビットが必要である．

Beckman ら [BCDP96] は，第 2 レジスタに $a^x \mod 15$ ではなく $\log_a(a^x \mod 15)$ を書き込むことでビット数を一般の場合には $\lceil \log_2 \log_a N \rceil$ ビットで Shor のアルゴリズムを実行できることを示した．ここでは，位数 4 の $a = 2$ のときの $N = 15$ の素因数分解を例に考え，第 2 レジスタのビット数を $\lceil \log_2 \log_2 15 \rceil = 2$ ビットとする．第 2 レジスタの冪乗演算前の状態は $|1\rangle$ であり，第 5.1 節の素朴な方法では量子状態 $|u\rangle$ に対して第 1 レジスタの制御ビットの値によって $\times 2 \mod 15$ と $\times 4 \mod 15$ を第 2 レジスタで計算する必要があった．Beckman ら [BCDP96] の方法では，量子状態 $|u\rangle$ に対して第 1 レジス

タの制御ビットの値によって $+1$ と $+2$ を第 2 レジスタで計算するすることで Shor のアルゴリズムを実行できる．つまり，第 2 レジスタの整数 $0, 2, 4, 8$ をそれぞれ $\log_a 1 = 0, \log_a 2 = 1, \log_a 4 = 2, \log_a 8 = 3$ として記録し，

$$U_2|u\rangle = |u \times 2 \mod 15\rangle,$$
$$U_4|u\rangle = |u \times 4 \mod 15\rangle$$

の演算を

$$U_{+1}|u\rangle = |u + 1 \mod 4\rangle,$$
$$U_{+2}|u\rangle = |u + 2 \mod 4\rangle$$

で置き換えている．また，厳密には $\mod 4$ が必要だが，高々 3 を足すだけなので $\mod 4$ は不要である．ただし，$\mod 4$ は不要であるというのは，$j \geq 2$ の $2^{2^j} \mod 15$ という冪乗演算が量子状態を変化させないという $N = 15$ の素因数分解に特化した話であり，位数が 4 で第 2 レジスタに足されうる最大の 3 より大きいという情報をも用いていることに注意されたい．また，ここで $\mod 4$ が必要になるのは，$N = 15$ のときの $a = 2$ の位数が $r = 4$ であることに起因するが，一般の合成数 N と a のペアにおいては r の値を用いずに計算をする必要があるため，同様の効率化を行うことはできない．

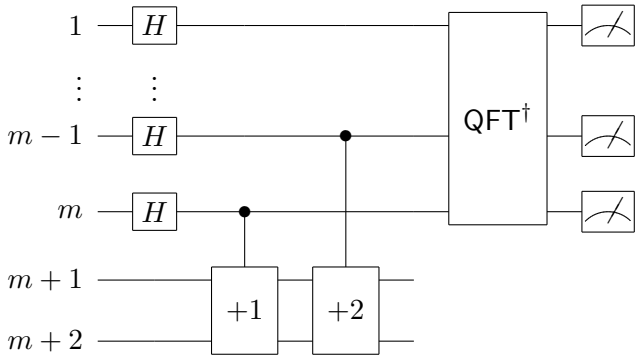

図 5.4: Beckman ら [BCDP96] の方法のもとで a の位数が $r = 4$ のときの $N = 15$ を素因数分解する量子回路

このように第 2 レジスタを置き換えたときの量子回路を図 5.4 に示す．このとき，位数計算アルゴリズムが適切に動作することを確認する．第 5.1 節では，$\times 2 \mod 15$ のユニタリ変換の固有ベクト

ルは

$$|u_0\rangle \coloneqq \frac{1}{\sqrt{4}}\sum_{k=0}^{3}|2^k \mod 15\rangle = \frac{|1\rangle+|2\rangle+|4\rangle+|8\rangle}{\sqrt{4}}$$
$$|u_1\rangle \coloneqq \frac{1}{\sqrt{4}}\sum_{k=0}^{3}e^{-\pi ik/2}|2^k \mod 15\rangle = \frac{|1\rangle-i|2\rangle-|4\rangle+i|8\rangle}{\sqrt{4}}$$
$$|u_2\rangle \coloneqq \frac{1}{\sqrt{4}}\sum_{k=0}^{3}e^{-\pi ik}|2^k \mod 15\rangle = \frac{|1\rangle-|2\rangle+|4\rangle-|8\rangle}{\sqrt{4}}$$
$$|u_3\rangle \coloneqq \frac{1}{\sqrt{4}}\sum_{k=0}^{3}e^{-3\pi ik/2}|2^k \mod 15\rangle = \frac{|1\rangle+i|2\rangle-|4\rangle-i|8\rangle}{\sqrt{4}}$$

の 4 つであり，

$$U_2|u_0\rangle = \frac{|2\rangle+|4\rangle+|8\rangle+|1\rangle}{\sqrt{4}} = |u_0\rangle$$
$$U_2|u_1\rangle = \frac{|2\rangle-i|4\rangle-|8\rangle+i|1\rangle}{\sqrt{4}} = e^{\pi i/2}|u_1\rangle$$
$$U_2|u_2\rangle = \frac{|2\rangle-|4\rangle+|8\rangle-|1\rangle}{\sqrt{4}} = e^{\pi i}|u_1\rangle$$
$$U_2|u_3\rangle = \frac{|2\rangle+i|4\rangle-|8\rangle-i|1\rangle}{\sqrt{4}} = e^{3\pi i/2}|u_1\rangle$$

より，それぞれの固有値は $1, e^{\pi i/2}, e^{\pi i}, e^{3\pi i/2}$ で，

$$\frac{|u_0\rangle+|u_1\rangle+|u_2\rangle+|u_3\rangle}{\sqrt{4}} = |1\rangle$$

という関係を満たすという条件によって位数計算アルゴリズムが動作していた．Beckman ら [BCDP96] の方法によって第 2 レジスタを書き換えると，4 つの固有ベクトルは

$$|u_0\rangle \coloneqq \ = \frac{|0\rangle+|1\rangle+|2\rangle+|3\rangle}{\sqrt{4}}$$
$$|u_1\rangle \coloneqq \ = \frac{|0\rangle-i|1\rangle-|2\rangle+i|3\rangle}{\sqrt{4}}$$
$$|u_2\rangle \coloneqq \ = \frac{|0\rangle-|1\rangle+|2\rangle-|3\rangle}{\sqrt{4}}$$

$$|u_3\rangle := = \frac{|0\rangle + i|1\rangle - |2\rangle - i|3\rangle}{\sqrt{4}}$$

となり，

$$U_{+1}|u\rangle \to |u+1 \mod 4\rangle$$

というユニタリ変換によって

$$\begin{aligned}
U_{+1}|u_0\rangle &= \frac{|1\rangle + |2\rangle + |3\rangle + |0\rangle}{\sqrt{4}} = |u_0\rangle \\
U_{+1}|u_1\rangle &= \frac{|1\rangle - i|2\rangle - |3\rangle + i|0\rangle}{\sqrt{4}} = e^{\pi i/2}|u_1\rangle \\
U_{+1}|u_2\rangle &= \frac{|1\rangle - |2\rangle + |3\rangle - |0\rangle}{\sqrt{4}} = e^{\pi i}|u_1\rangle \\
U_{+1}|u_3\rangle &= \frac{|1\rangle + i|2\rangle - |3\rangle - i|0\rangle}{\sqrt{4}} = e^{3\pi i/2}|u_1\rangle
\end{aligned}$$

となり，それぞれの固有値は $1, e^{\pi i/2}, e^{\pi i}, e^{3\pi i/2}$ である．また，これらの固有ベクトルは

$$\frac{|u_0\rangle + |u_1\rangle + |u_2\rangle + |u_3\rangle}{\sqrt{4}} = |0\rangle$$

という関係を満たし，第 2 レジスタの初期状態を $|0\rangle$ とすることで位数計算アルゴリズムを実行できる．ただし，4 つの固有ベクトルを

$$\begin{aligned}
|u_0\rangle &:= \frac{|1\rangle + |2\rangle + |3\rangle + |0\rangle}{\sqrt{4}} \\
|u_1\rangle &:= \frac{|1\rangle - i|2\rangle - |3\rangle + i|0\rangle}{\sqrt{4}} \\
|u_2\rangle &:= \frac{|1\rangle - |2\rangle + |3\rangle - |0\rangle}{\sqrt{4}} \\
|u_3\rangle &:= \frac{|1\rangle + i|2\rangle - |3\rangle - i|0\rangle}{\sqrt{4}}
\end{aligned}$$

としたとき，

$$U_{+1}|u_0\rangle = \frac{|2\rangle + |3\rangle + |0\rangle + |1\rangle}{\sqrt{4}} = |u_0\rangle$$

$$U_{+1}|u_1\rangle = \frac{|2\rangle - i|3\rangle - |0\rangle + i|1\rangle}{\sqrt{4}} = e^{\pi i/2}|u_1\rangle$$
$$U_{+1}|u_2\rangle = \frac{|2\rangle - |3\rangle + |0\rangle - |1\rangle}{\sqrt{4}} = e^{\pi i}|u_1\rangle$$
$$U_{+1}|u_3\rangle = \frac{|2\rangle + i|3\rangle - |0\rangle - i|1\rangle}{\sqrt{4}} = e^{3\pi i/2}|u_1\rangle$$

となり，それぞれの固有値は $1, e^{\pi i/2}, e^{\pi i}, e^{3\pi i/2}$ であり，

$$\frac{|u_0\rangle + |u_1\rangle + |u_2\rangle + |u_3\rangle}{\sqrt{4}} = |1\rangle$$

が成り立つ．よって，第 2 レジスタの初期状態を $|1\rangle$ としても同様に素因数分解のための位数計算アルゴリズムを実行可能である．

5.2.2　半古典フーリエ変換

Shor のアルゴリズムでは，半古典フーリエ変換を用いることで計算時間が増える代わりに必要な量子ビット数を減らせることがわかっている．本節で半古典フーリエ変換を紹介するが，まずは量子フーリエ変換について復習する．

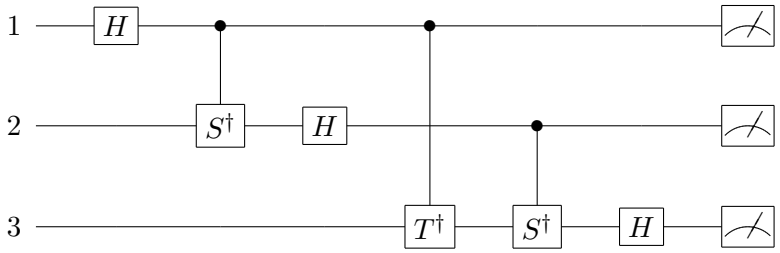

図 5.5: 3 ビット量子逆フーリエ変換の量子回路

図 5.5 は，3 ビットの量子逆フーリエ変換の量子回路である．この量子回路に

$$\frac{|0\rangle + e^{2\pi i \cdot 0.j_0}|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + e^{2\pi i \cdot 0.j_1 j_0}|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + e^{2\pi i \cdot 0.j_2 j_1 j_0}|1\rangle}{\sqrt{2}}$$

を入力したとき，第 1 ビットの Hadamard ゲートによって第 1 量子ビットの量子状態は

$$H\frac{|0\rangle + e^{2\pi i\cdot 0.j_0}|1\rangle}{\sqrt{2}} = \frac{1}{2}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}1\\ e^{2\pi i\cdot 0.j_0}\end{pmatrix}$$
$$= |j_0\rangle$$

となる．$j_0 = 0$ のとき，第 2 ビット目の制御 S ゲートは計算されないので Hadamard ゲートのみが作用し，同様にして第 2 量子ビットの量子状態は $|j_1\rangle$ となる．また，$j_0 = j_1 = 0$ のときには第 3 量子ビットの量子状態が $|j_2\rangle$ となることも同様に確認できる．$j_0 = 1$ のとき，制御 S ゲートによって第 2 量子ビットの量子状態は

$$S^\dagger\frac{|0\rangle + e^{2\pi i\cdot 0.j_1 1}|1\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 0\\ 0 & -i\end{pmatrix}\begin{pmatrix}1\\ e^{2\pi i\cdot 0.j_1}\cdot e^{\pi i/2}\end{pmatrix}$$
$$= \frac{1}{\sqrt{2}}\begin{pmatrix}1\\ e^{2\pi i\cdot 0.j_1}\end{pmatrix}$$
$$= \frac{|0\rangle + e^{2\pi i\cdot 0.j_1}|1\rangle}{\sqrt{2}}$$

となる．よって，Hadamard ゲートをかけた後に第 2 量子ビットの量子状態が $|j_1\rangle$ となることがわかる．同様にして，$j_0 = 0, j_1 = 1$ のときには第 3 量子ビットの量子状態は制御 $S^\dagger$ ゲートと Hadamard ゲートの後に $|j_2\rangle$ となることがわかる．さらに，$j_0 = 1$ のときには制御 $T^\dagger$ の後の第 3 量子ビットの量子状態は

$$T^\dagger\frac{|0\rangle + e^{2\pi i\cdot 0.j_2 j_1 1}|1\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 0\\ 0 & e^{-\pi i/4}\end{pmatrix}\begin{pmatrix}1\\ e^{2\pi i\cdot 0.j_2 j_1}\cdot e^{\pi i/4}\end{pmatrix}$$
$$= \frac{1}{\sqrt{2}}\begin{pmatrix}1\\ e^{2\pi i\cdot 0.j_2 j_1}\end{pmatrix}$$
$$= \frac{|0\rangle + e^{2\pi i\cdot 0.j_2 j_1}|1\rangle}{\sqrt{2}}$$

となる．よって，これまでと同様にして，制御 S ゲートと Hadamard ゲートの後には第 3 量子ビットの量子状態は $|j_2\rangle$ となることがわかる．これによって，図 5.5 の量子回路が 3 ビットの量子逆フーリエ変換のものであることが確認できた．

図 5.5 の 3 ビットは，位数計算アルゴリズムの中では最後に量子逆フーリエ変換をかける第 1 レジスタのものに相当する．図 5.5 のとおり，量子逆フーリエ変換の量子回路はいくつかの 2 ビットゲー

トによって構成されており，2 ビットゲートは 1 ビットゲートに比べて物理的に構成が難しい．この問題を解決するため，フーリエ変換を 1 ビットゲートのみを用いて半古典に実行する方法が知られている [GN93]．

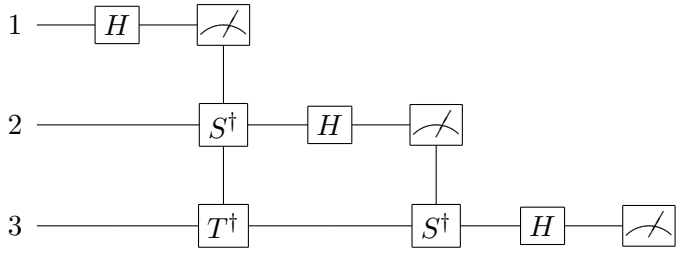

図 5.6: 3 ビット半古典逆フーリエ変換の量子回路

図 5.6 は，3 ビットの半古典フーリエ変換の量子回路を表したものである．図 5.5 の量子回路とは異なり，図 5.6 では先に第 1 ビットを測定することで得られた古典情報によって第 2 ビットと第 3 ビットにそれぞれ $S^\dagger, T^\dagger$ ゲートをかけるかを決定している．同様に，先に第 2 ビットを測定することで得られた古典情報によって第 3 ビットに $S^\dagger$ ゲートをかけるかを決定している．これによって，図 5.5 の量子回路における全ての 2 ビットゲートを図 5.6 では 1 ビットゲートに置き換えることができている．さらに，量子ビットリサイクリングという手法によって，一般的に第 1 レジスタのビット数を 1 として位数計算アルゴリズムを実行することができる．例として，図 5.3 の量子回路を半古典フーリエ変換と量子ビットリサイクリングを用いて書き換えた量子回路が図 5.7 となる．ここで，第 2 レジスタを 4 倍した後の 1 ビットゲート $S^\dagger$ と第 2 レジスタを a 倍した後の 1 ビットゲート $T^\dagger$ は，最初の測定で 1 を観測したときのみ行われ，また，第 2 レジスタを a 倍した後の 1 ビットゲート $S^\dagger$ は，2 回目の測定で 1 を観測したときのみ行われるものである．ただし，紙面の都合上図 5.3 の量子回路に半古典フーリエ変換と量子ビットリサイクリングを適用した回路を，第 1 レジスタが常に第 1 ビットであることを明記して図 5.8 のように書くことにする．

5.3 既存の実装結果

本節で既存の Shor のアルゴリズムの実験の結果についてまとめる．これらは，[VSB+01, LWL+07, LBYP07, PMO09, LBC+12, MLLL+12, SSV13, MNM+16, ASK19, DLQ+20] による素因数分解実験と，[AST+20] による素体上の離散対数実験からなる．表 5.1 にこれまでの素因数分解の実験結

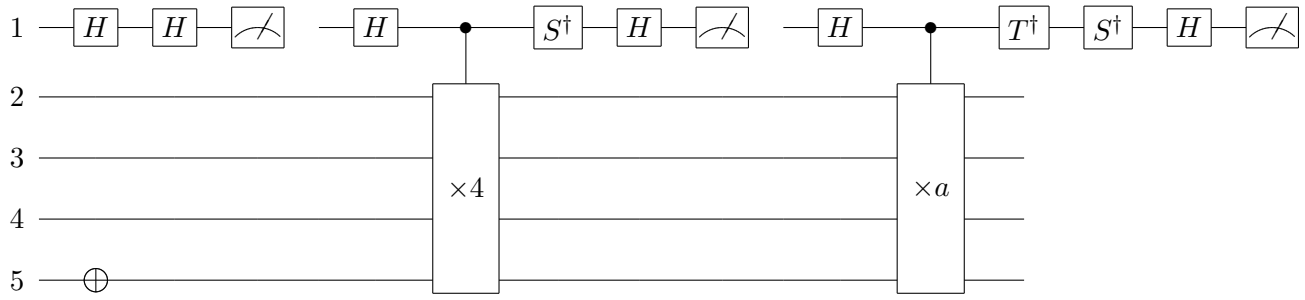

図 5.7: 半古典フーリエ変換と量子ビットリサイクリングを用いたとき a の位数が 4 のときの 15 を素因数分解する量子回路

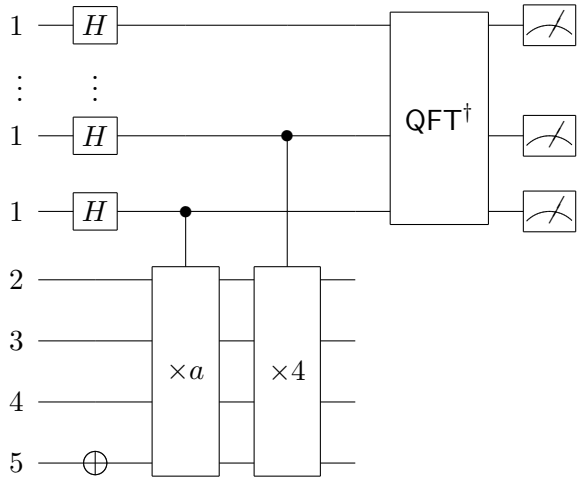

図 5.8: 半古典フーリエ変換と量子ビットリサイクリングを用いたとき a の位数が 4 のときの 15 を素因数分解する量子回路の簡略図

果をまとめている．これまで素因数分解されたのは，いずれも小さな合成数 $N = 15, 21$ のみであり，[AST+20] の離散対数問題実験も $2^x = 1 \mod 3$ という非常に小さなパラメータにおける場合である．[SSV13] の実験は大きな合成数まで素因数分解可能としているが，これは一般的な素因数分解アルゴリズムとはなっていない．詳細については第 5.3.7 小節で確認する． [MLLL+12] の量子ビット数が $1 + \log_2 3$ となっているが，この実験においては $|0\rangle$ または $|1\rangle$ の 2 つの量子状態の重ね合わせである量子ビットと $|0\rangle, |1\rangle, |2\rangle$ の 3 つの量子状態の重ね合わせであるキュートリット（qutrit）を用いているためである．また，同じ合成数 $N = 15$ を素因数分解するにも量子ビット数や冪乗演算のゲート数が異なっているが，ここでの効率化は一般の素因数分解に拡張できるものではなく，$N = 15$ の素因分解に特化したものが多い．そのため，冪乗演算のゲート数が多い実験は，過度に $N = 15, 21$ の場合に特化せずに一般的な量子回路に近い実装をしていると言える．

これより，第 5.3.1 小節から第 5.3.10 小節で既存の素因数分解実験についてまとめ，第 5.3.11 小節で [AST+20] の素体上の離散対数実験についてまとめる．

5.3.1 [VSB+01] の実験

Vandersypen ら [VSB+01] は，位数が 2 である $a = 11$ と 4 である $a = 7$ のときに 15 の素因数分解実験を行なった．

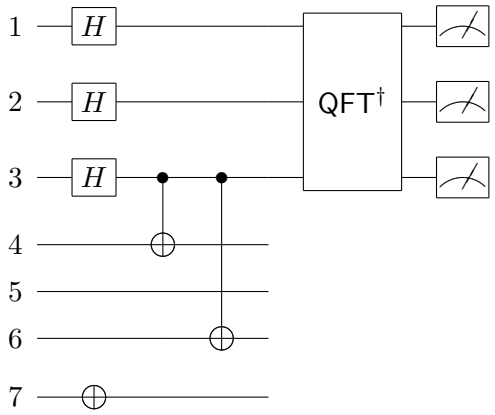

図 5.9: $a = 11$ の位数が 2 のときの 15 を素因数分解する [VSB+01] の量子回路

図 5.9 は，Vandersypen ら [VSB+01] が $a = 11$ のときに $N = 15$ を素因数分解するために用いた量子回路である．第 1 レジスタは 3 ビットであり，第 2 レジスタは 15 以下の正整数の値を書き込め

表 5.1: Shor の素因数分解アルゴリズムの実装実験結果

研究グループ	合成数	a	量子ビット数	冪乗演算のゲート数	逆フーリエ変換
[VSB+01]	15	11 7	7	2 9	量子
[LWL+07]	15	4 2	7 5	2 2	量子
[LBYP07]	15	11	5	2	量子
[PMO09]	15		5	6	量子
[LBC+12]	15	4	3	2	量子
[MLLL+12]	21	4	$1 + \log_2 3$		半古典
[SSV13]	15 RSA-768 N-20000		2	1	半古典
[MNM+16]	15	2 7 8 11 13	5	11 15 11 2 15	半古典
[ASK19]	15 15 21	2 11 2	5 5 6	2 8 19	半古典
[DLQ+20]	15	7, 8	3	3	半古典

るように 4 ビットである．ここで，冪乗演算のために 11 倍を計算する必要があるが，図 5.1 のような素朴な回路を用いていない．Vandersypen ら [VSB+01] は，1 を 11 倍する計算を 1 に 10 を足す計算に置き換えることで回路を簡略化している．実際，二つの固有ベクトル $|u_0\rangle$ と $|u_1\rangle$ は，このように計算を置き換えても式 (5.1) が成り立つことがわかる．これにより，制御ビットが 0 と 1 のときに第 2 レジスタはそれぞれ 1 と 11 になるので，所望の計算を簡略化された回路で実現できている．これは，演算前の第 2 レジスタの値が 1 であるという事実を利用している．このように，最初の演算を簡略化することは一般の素因数分解においても可能だと考えられるが，この簡略化は 1, 2 回の乗算が必要な 15 の素因数分解では非常に有効であるが，$m = O(n)$ 回の乗算が必要な一般の素因数分解ではさほど効果はないと考えられる．

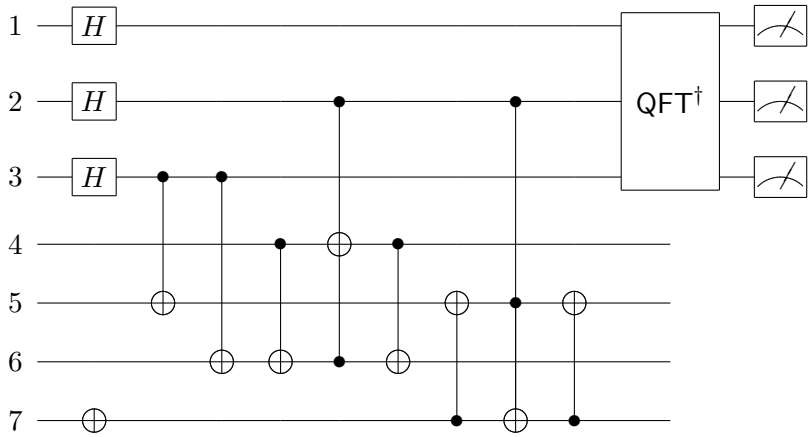

図 5.10: $a = 7$ の位数が 4 のときの 15 を素因数分解する簡略化前の [VSB+01] の量子回路

Vandersypen ら [VSB+01] が $a = 7$ のときに $N = 15$ を素因数分解するために用いた量子回路は図 5.11 であるが，まず図 5.10 の量子回路を見てみる．第 1 レジスタは 3 ビットであり，第 2 レジスタは 15 以下の正整数の値を書き込めるように 4 ビットである．図 5.11 の量子回路は図 5.10 の量子回路をより簡略化したものとなっている．図 5.10 において，冪乗演算のために 7 倍と 4 倍を計算する必要があるが，図 5.9 のときと同様に 1 を入力として 7 倍する回路を二つの制御 NOT ゲートによって 6 を足す回路に簡略化している．これにより，制御ビットが 0 と 1 のときに第 2 レジスタはそれぞれ 1 と 7 になるので，所望の計算を簡略化された回路で実現できている．さらに，4 倍する回路も簡略化されている．図 5.10 で 4 倍を計算するための 2 つの Toffoli ゲートと 1 つの NOT ゲートは，簡略化されていなければ図 5.11 のように 4 つの制御 NOT ゲートと 2 つの Toffoli ゲートからなる．

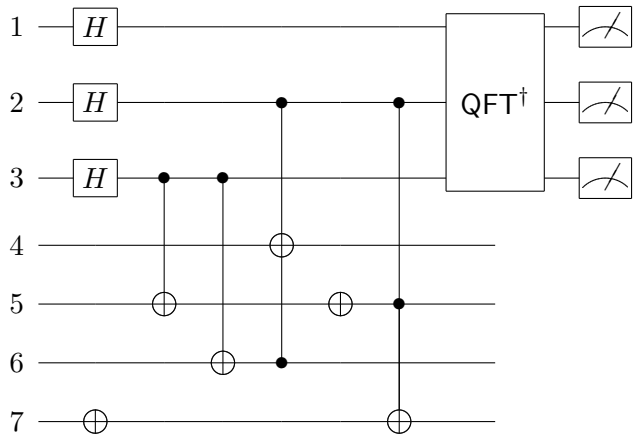

図 5.11: $a = 7$ の位数が 4 のときの 15 を素因数分解する簡略化後の [VSB+01] の量子回路

さらに，図 5.10 の量子回路をより簡略化すると図 5.11 の量子回路となることを確認する．図 5.10 の量子回路において，4 ビット目は常に 0 のままである．よって，4 ビット目を制御ビットとして 6 ビット目を標的ビットとする二つの制御 NOT ゲートは省略することができる．また，7 ビット目を制御ビットとする一つ目の制御 NOT ゲートは，このときには 7 ビット目が常に 1 であることから NOT ゲートに置き換えることができる．さらに，最後の 7 ビット目を制御ビットとする制御 NOT ゲートは，第 1 レジスタの状態に影響を与えないので省略することができる．これにより，図 5.10 の量子回路で所望の計算が可能であることがわかる．

5.3.2 [LWL+07] の実験

Lanyon ら [LWL+07] は，位数が 2 である $a = 4$ と 4 である $a = 2$ のときに 15 の素因数分解実験を行った．

図 5.12 は，Lanyon ら [LWL+07] が $a = 4$ のときに $N = 15$ を素因数分解するために用いた量子回路である．第 1 レジスタは 3 ビットであり，第 2 レジスタは 15 以下の正整数の値を書き込めるように 4 ビットである．この量子回路は，本質的には Vandersypen ら [VSB+01] が $a = 11$ のときに用いた回路である図 5.9 と同じである．つまり，冪乗演算のために 4 倍を計算する必要があるが，図 5.1 のような素朴な回路ではなく，1 に 3 を足す回路に置き換えている．これにより，制御ビットが 0 と

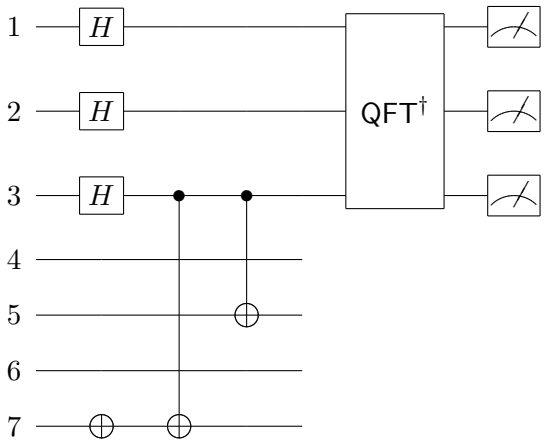

図 5.12: $a = 4$ で 15 を素因数分解する [LWL+07] の量子回路

1 のときに第 2 レジスタはそれぞれ 1 と 4 になるので，所望の計算を簡略化された回路で実現できている．これは，演算前の第 2 レジスタの値が 1 であるという事実を利用している．

Lanyon ら [LWL+07] が $a = 2$ のときに $N = 15$ を素因数分解するために用いた量子回路は図 5.13 である．この量子回路は，第 5.2.1 で紹介した Beckman ら [BCDP96] による第 2 レジスタを圧縮した量子回路である図 5.4 の量子回路である．これによって，$a = 4$ のときより少ない量子ビット数で素因数分解に成功している．

5.3.3 [LBYP07] の実験

Lu ら [LBYP07] は，位数が 2 である $a = 11$ のときに 15 の素因数分解実験を行った．

Lu ら [LBYP07] が $a = 11$ のときに $N = 15$ を素因数分解するために用いた量子回路は図 5.15 であるが，まず図 5.14 の量子回路を見てみる．図 5.14 の量子回路は本質的には Vandersypen ら [VSB+01] の $a = 11$ のときの量子回路である図 5.9 と等しいが，第 1 レジスタは 2 ビットとなっており，その分だけ簡略化されている．

さらに，図 5.15 のように，Lu ら [LBYP07] は量子ビットリサイクリングと半古典フーリエ変換を用いており，第 1 量子ビットをエンタングル状態にすることなく実験を行なっている．

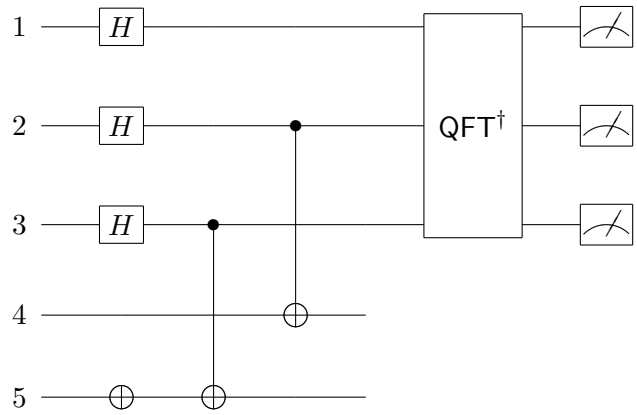

図 5.13: $a = 2$ で 15 を素因数分解する [LWL+07] の簡略化後の量子回路

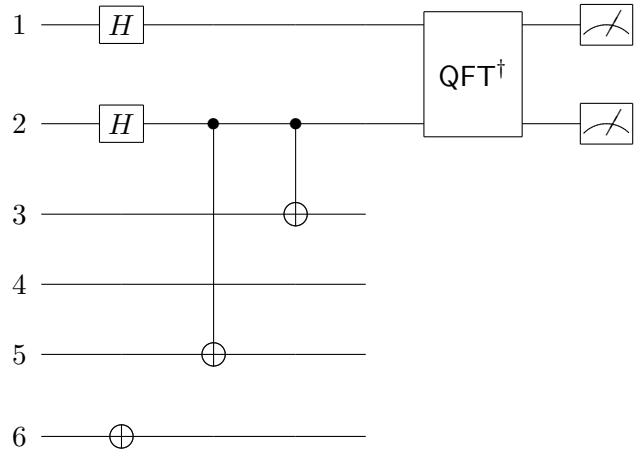

図 5.14: $a = 11$ で 15 を素因数分解する [LBYP07] の簡略化する前の量子回路

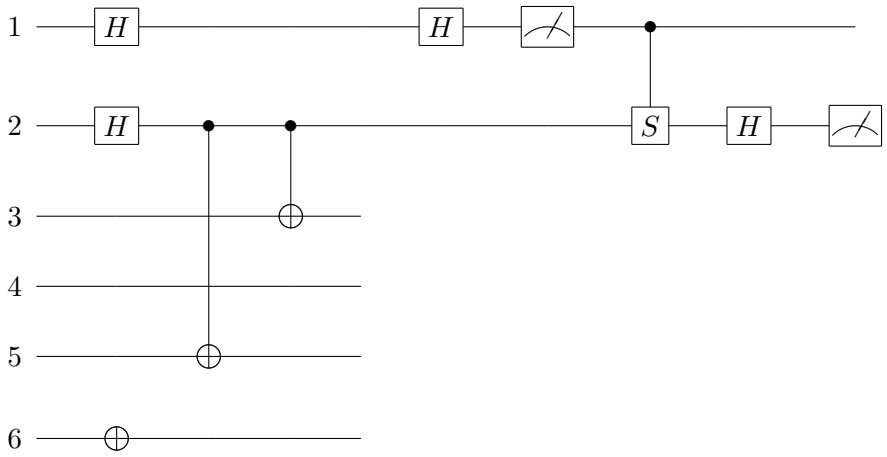

図 5.15: $a = 11$ で 15 を素因数分解する [LBYP07] の簡略化した量子回路

5.3.4 [PMO09] の実験

Politi ら [PMO09] は，位数が 4 である a を用いて 15 の素因数分解実験を行なった．

図 5.16 は Politi らの量子回路の概要である．論文中に a の値が明記されていないが，この回路は Lanyon ら [LWL+07] による図 5.13 の量子回路と同じである．

また，Politi ら [PMO09] は制御 NOT 演算を 2 つの Hadamard ゲートと制御 Z ゲートによって実現しており，実験に用いた量子回路は図 5.17 である．

5.3.5 [LBC+12] の実験

Lucero ら [LBC+12] は，位数が 2 である $a = 4$ のときに 15 の素因数分解実験を行った．

Lucero ら [LBC+12] が $a = 4$ のときに $N = 15$ を素因数分解するために用いた量子回路は図 5.19 であるが，まず図 5.18 の量子回路を見てみる．図 5.18 の量子回路は，図 5.12 の Lanyon ら [LWL+07] の量子回路の第 1 レジスタを 2 ビットにしたものであり，全体では 6 量子ビットを用いている．

図 5.18 の量子回路を簡略化し，Lucero ら [LBC+12] は図 5.19 の量子回路で実験を行なった．このとき，図 5.18 の 3 ビット目と 5 ビット目は演算を行わないので省略している．さらに，第 1 ビットは入力 $|0\rangle$ に Hadamard ゲートを 2 回かけるだけなので，常に $|1\rangle$ となり冗長である．よって，第 1

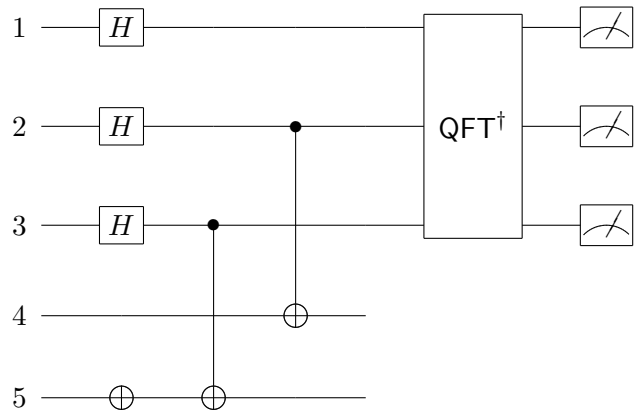

図 5.16: 15 を素因数分解する [PMO09] の量子回路の概要

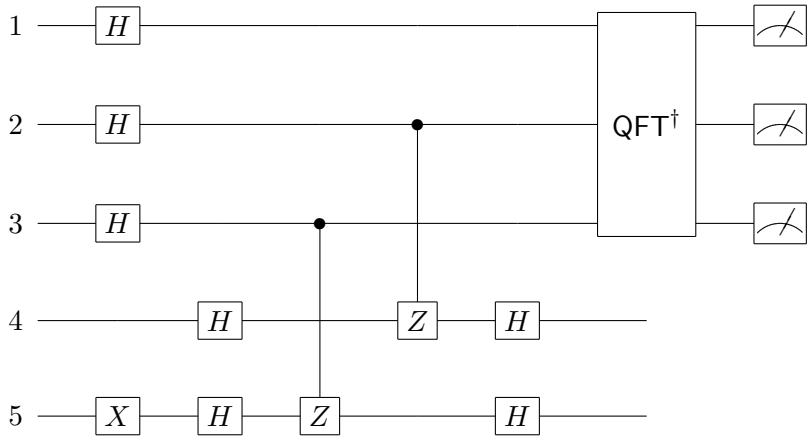

図 5.17: 15 を素因数分解する [PMO09] の量子回路

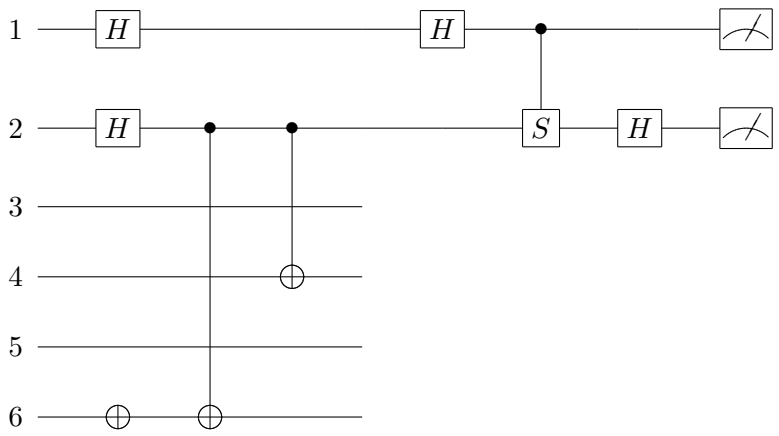

図 5.18: $a = 4$ で 15 を素因数分解する [LBC+12] の簡略化する前の量子回路

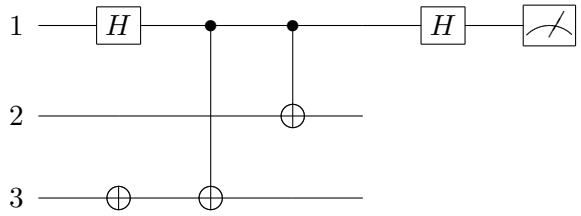

図 5.19: $a = 4$ で 15 を素因数分解する [LBC+12] の量子回路

ビットを省略して第 1 レジスタを 1 ビットとしている．これらの簡略化により，図 5.19 の量子回路を得る．

5.3.6 [MLLL+12] の実験

Martín-López ら [MLLL+12] は，位数が 3 である $a = 4$ のときに 21 の素因数分解実験を行った．第 4 章で述べたように，Shor のアルゴリズムでは基本的に位数が偶数である必要がある．ただし，$a = 4$ が平方数であることより，$\gcd(4^{3/2} \pm 1, 21)$ によって 21 の素因数 3 と 7 を計算することができる．

Martín-López ら [MLLL+12] の実装では，量子ビットリサイクリングと半古典フーリエ変換を用いることで，第 1 レジスタを 1 ビットとしている．また，素朴に実装すると第 2 レジスタには 0 から 20 までの非負整数を表現するために 5 ビットが必要となるが，Martín-López ら [MLLL+12] の実装の最大の特徴として，第 2 レジスタには一つのキュートリット（qutrit）を用いている．量子ビットが $|0\rangle$ と $|1\rangle$ の二つの量子状態の重ね合わせを表現するのに対して，キュートリットは $|0\rangle, |1\rangle, |2\rangle$ の三つの量子状態の重ね合わせを表現できる．これによって，全体として $1 + \log_2 3$ 量子ビットで実験を行なっている．第 2 レジスタの圧縮について発想は第 5.2.1 節と近いが，Martín-López らの実装が成功するのには，位数が 3 以下であるという情報を陽に用いていることに注意されたい．ここでは，$\log_4(4 \mod 21) = 1, \log_4(4^2 \mod 21) = 2, \log_4(4^3 \mod 21) = 0$ であることを利用して，$|1\rangle, |2\rangle, |0\rangle$ をそれぞれ割り当てている．キュートリットを用いた位数計算アルゴリズムはこれまで説明したものと方法が異なるため，ここでは詳細は省略する．

5.3.7 [SSV13] の実験

Smolin ら [SSV13] は，位数が 2 である a がわかれば，任意の N を 2 量子ビットの量子回路で素因数分解可能であることを示した．

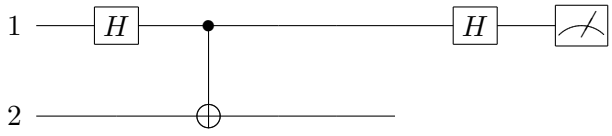

図 5.20: 位数 2 の a によって素因数分解する [SSV13] の量子回路

図 5.20 がそのときの量子回路である．この量子回路は，第 5.2.1 で紹介した Beckman ら [BCDP96] による第 2 レジスタを圧縮した量子回路である図 5.4 と本質的に等しい．つまり，図 5.4 の回路では位数が 4 の a を考えているため，$+1, +2$ の演算が必要となるが，位数が 2 の場合に限定すれば，やるべき演算は $+1$ のみで済むため，図 5.20 のとおり一つの制御 NOT ゲートで十分である．

Smolin ら [SSV13] によると，位数が 2 である a がわかれば，図 5.20 の量子回路によって $N = 15$ のような小さな合成数のみならず，RSA-768 や 20000 ビットの合成数である N-20000 さえも素因数分解可能であるとした．この論文の意図としては，そのような大きな合成数ですら素因数分解可能であるということではなく，a の位数がわかっている場合に量子回路を大幅に簡略化した実装実験は，本質的には将来実用的に素因数分解を行うことには繋がらないということを示唆している．

5.3.8 [MNM+16] の実験

Monz ら [MNM+16] は，位数が 2 である $a = 11$ と位数が 4 である $a = 2, 7, 8, 13$ のときに 15 の素因数分解実験を行った．

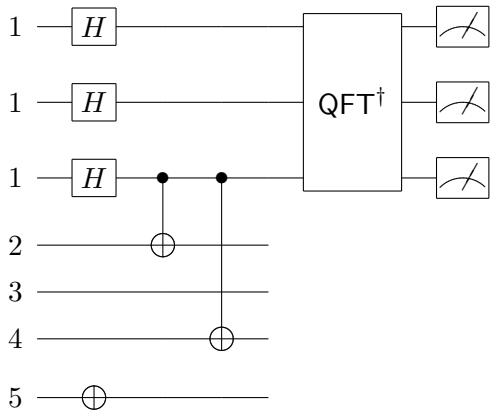

図 5.21: $a = 11$ の位数が 2 のときの 15 を素因数分解する [MNM+16] の量子回路

図 5.21 は，Monz ら [MNM+16] が位数が 2 である $a = 11$ のときに用いた量子回路である．この量子回路は，本質的には Vandersypen ら [VSB+01] の図 5.9 と等しい．ただし，Monz らは半古典フーリエ変換を用いることで第 1 レジスタを 1 ビットとしている．

図 5.21 は，Monz ら [MNM+16] が位数が 4 である $a = 2, 7, 8, 13$ のときに用いた量子回路である．いずれの場合も $a^2 \mod 15 = 4$ であることより，最初の 4 倍の計算を二つの制御 NOT ゲートによっ

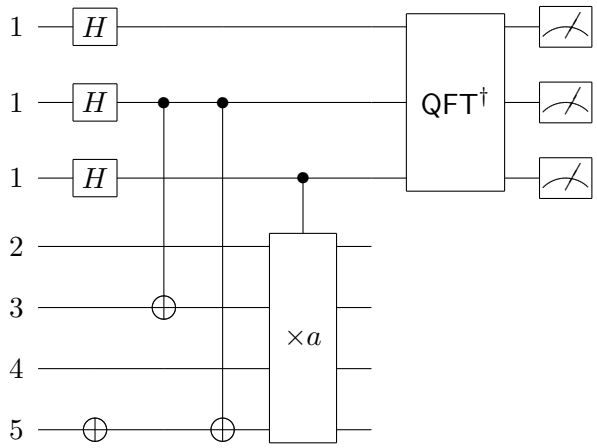

図 5.22: $a = 2, 7, 8, 13$ の位数が 4 のときの 15 を素因数分解する [MNM+16] の量子回路

て計算している．その後の a 倍の計算は，それぞれ $a = 2, 7, 8, 13$ のときの図 5.1 の量子ゲートによって行なっている．つまり，これまで説明した他の実験と比べると必要以上に $N = 15$ であることに特化した簡略化を行なっていない．Monz らは半古典フーリエ変換を用いており，第 1 レジスタを 1 ビットとしていることに注意されたい．

5.3.9 [ASK19] の実験

Amico ら [ASK19] は，位数が 2 である $a = 11$ と位数が 4 である $a = 2$ のときの 15 の素因数分解実験，位数が 6 である $a = 2$ のときの $N = 21$ の素因数分解を行なった．

図 5.23 は，Amico ら [ASK19] が位数が 2 である $a = 11$ のときに $N = 15$ を素因数分解した量子回路である．これは，Monz ら [MNM+16] による図 5.21 の量子回路と同じである．

図 5.24 は，Amico ら [ASK19] が位数が 4 である $a = 2$ のときに $N = 15$ を素因数分解した量子回路である．これは，Monz ら [MNM+16] による図 5.22 とよく似ている．ただし，Monz らが 2 倍をするために用いた量子ゲートは図 5.1 のものであり，図 5.24 では第 3 ビットと第 4 ビットをスワップする量子ゲートが省略されている．ここで Amico らは，2 倍をする直前での第 2 レジスタの量子状態が $|1\rangle$ または $|4\rangle$ のいずれかであり，第 3 ビットと第 4 ビットをスワップする操作が発生しないことを利用して簡略化している．

図 5.25 は，Amico ら [ASK19] が位数が 6 である $a = 2$ のときに $N = 21$ を素因数分解した量子回

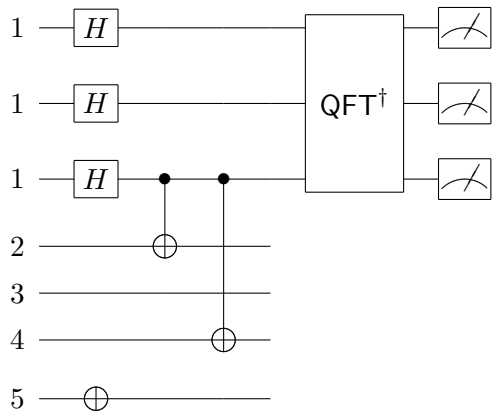

図 5.23: $a = 11$ の位数が 2 のときの 15 を素因数分解する [ASK19] の量子回路

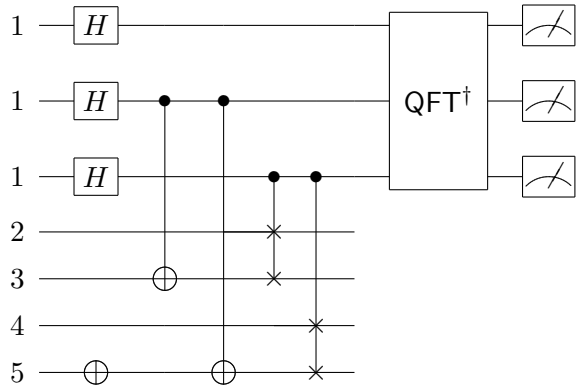

図 5.24: $a = 2$ の位数が 4 のときの 15 を素因数分解する [ASK19] の量子回路

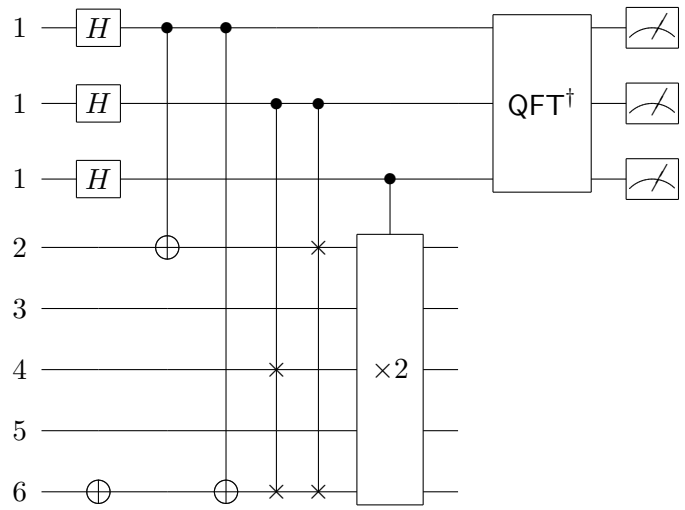

図 5.25: $a = 2$ の位数が 6 のときの 21 を素因数分解する [ASK19] の量子回路

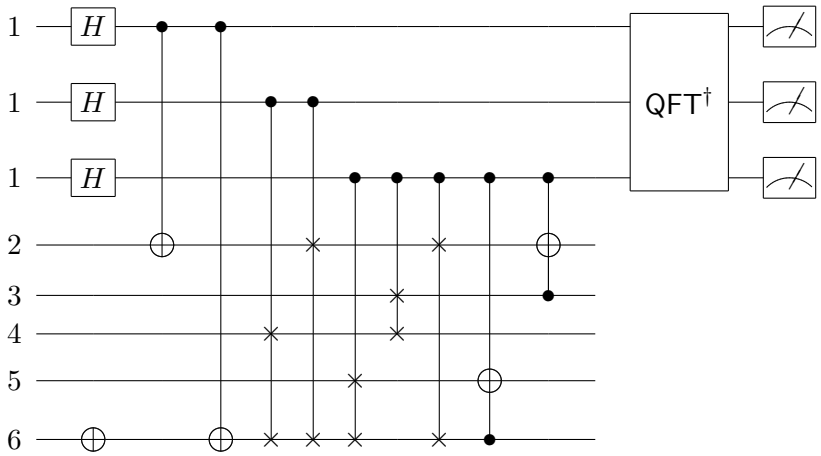

図 5.26: 測定が $00, 10, 01$ で $a = 2$ の位数が 6 のときの 21 を素因数分解する [ASK19] の量子回路

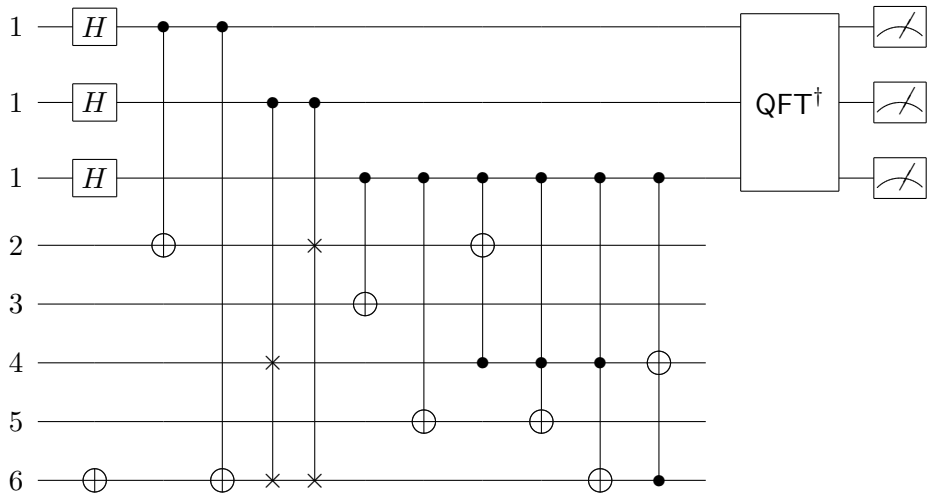

図 5.27: 測定が 11 で $a = 2$ の位数が 6 のときの 21 を素因数分解する [ASK19] の量子回路

路である．二つの制御 NOT ゲートで $2^4 \mod 21$ 倍の演算を行い，二つの制御スワップゲートで $2^2 \mod 21$ 倍の演算を行なっている．ここでは詳細は省略するが，半古典フーリエ変換による観測結果に応じて $2 \mod 21$ の演算に異なる量子ゲートを用いており，詳細は図 5.26，5.27 に示す．

また，Amico ら [ASK19] は図 5.28 の量子回路を用いて位数が 6 である $a = 4$ のときの $N = 35$ の素因数分解をも試みているが，観測される値がほぼランダムで実験は成功しなかったとしている．

5.3.10 [DLQ+20] の実験

Duan ら [DLQ+20] は，位数が 4 である $a = 7, 8$ のときに 15 の素因数分解実験を行った．

図 5.29 は Duan らが用いた量子回路である．ただし，実際の実験では半古典フーリエ変換と量子ビットリサイクリングを用いていることに注意されたい．この量子回路では，第 5.2.1 で紹介した Beckman ら [BCDP96] による第 2 レジスタを圧縮を用いている．ただし，第 2 レジスタで計算しているのは，$\log_a(a^x \mod 15)$ ではなく，$\log_2((-1)^{ax} \cdot a^x \mod 15)$ である．$a = 7, 8$ のときに量子回路が等しいことは，$(-1)^7 \cdot 7 \mod 15 = 8 = (-1)^8 \cdot 8 \mod 15$ と $(-1)^{14} \cdot 7^2 \mod 15 = 4 = (-1)^{16} \cdot 8^2 \mod 15$ から確認できる．

第 2 レジスタでの計算が第 5.2.1 のときとは異なるため，図 5.29 のように $\log_2((-1)^{ax} \cdot a^x \mod 15)$ としても位数計算アルゴリズムが成功することを $a = 7$ の場合を用いて簡単に確認する．4 つの固有

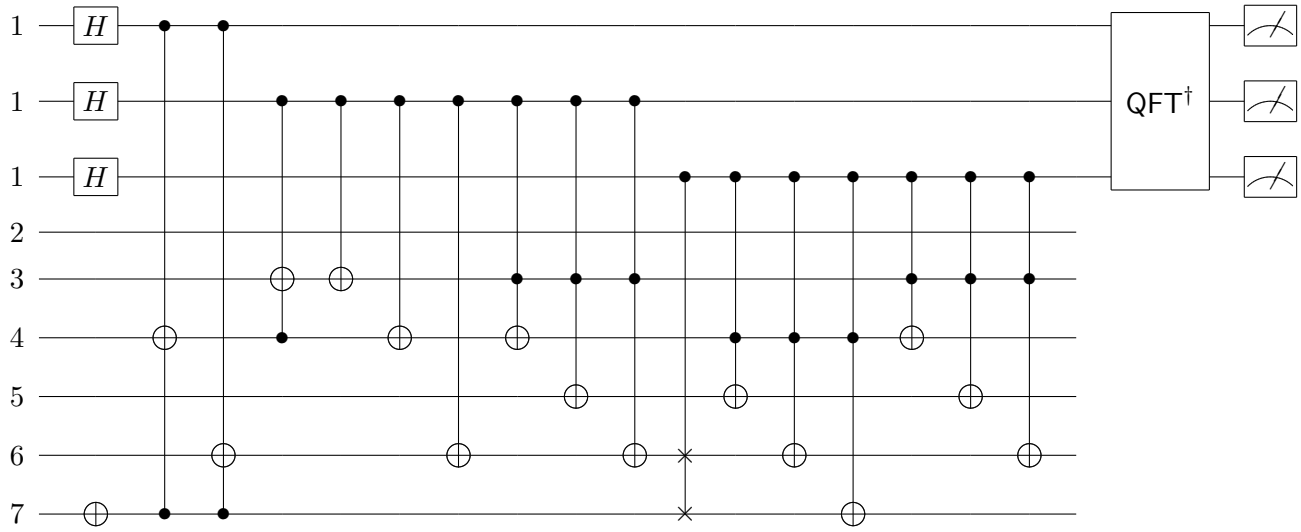

図 5.28: $a = 4$ の位数が 6 のときの 35 を素因数分解する [ASK19] の量子回路

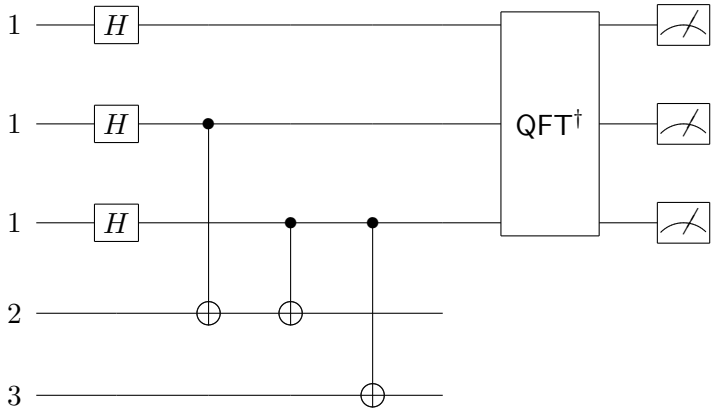

図 5.29: $a = 7, 8$ で 15 を素因数分解する [DLQ+20] の量子回路の概要

ベクトルを

$$
\begin{aligned}
|u_0\rangle &:= \ = \frac{|0\rangle + |1\rangle + |2\rangle + |3\rangle}{\sqrt{4}} \\
|u_1\rangle &:= \ = \frac{|0\rangle - i|1\rangle - |2\rangle + i|3\rangle}{\sqrt{4}} \\
|u_2\rangle &:= \ = \frac{|0\rangle - |1\rangle + |2\rangle - |3\rangle}{\sqrt{4}} \\
|u_3\rangle &:= \ = \frac{|0\rangle + i|1\rangle - |2\rangle - i|3\rangle}{\sqrt{4}}
\end{aligned}
$$

としたとき，$\log_2((-1)^7 \cdot 7 \mod 15) = 3$ であることにより，

$$U_{+3}|u\rangle \to |u+3 \mod 4\rangle$$

というユニタリ変換によって

$$
\begin{aligned}
U_{+3}|u_0\rangle &= \frac{|3\rangle + |0\rangle + |1\rangle + |2\rangle}{\sqrt{4}} = |u_0\rangle \\
U_{+3}|u_1\rangle &= \frac{|3\rangle - i|0\rangle - |1\rangle + i|2\rangle}{\sqrt{4}} = e^{3\pi i/2}|u_1\rangle \\
U_{+3}|u_2\rangle &= \frac{|3\rangle - |0\rangle + |1\rangle - |2\rangle}{\sqrt{4}} = e^{\pi i}|u_1\rangle \\
U_{+3}|u_3\rangle &= \frac{|3\rangle + i|0\rangle - |1\rangle - i|2\rangle}{\sqrt{4}} = e^{\pi i/2}|u_1\rangle
\end{aligned}
$$

となり，それぞれの固有値は $1, e^{3\pi i/2}, e^{\pi i}, e^{\pi i/2}$ である．また，これらの固有ベクトルは

$$\frac{|u_0\rangle + |u_1\rangle + |u_2\rangle + |u_3\rangle}{\sqrt{4}} = |0\rangle$$

という関係を満たし，第 2 レジスタの初期状態を $|0\rangle$ とすることで位数計算アルゴリズムを実行できる．この関係は $a = 8$ のときのも同様であることがわかる．

5.3.11 [AST+20] の離散対数実験

青野ら [AST+20] は，$2^x = 1 \mod 3$ を解く素体上の離散対数問題の実験を行った．このとき，青野らは 6 量子ビットと 7 量子ビットの量子回路を用いたが，それぞれを図 5.30 と図 5.31 に示す．さらに，青野らは $2^x = 2 \mod 3, 4^x = 2 \mod 7, 3^x = 4 \mod 7$ の場合の実験も試みたが，これらの場合には実験が成功しなかったとしている．

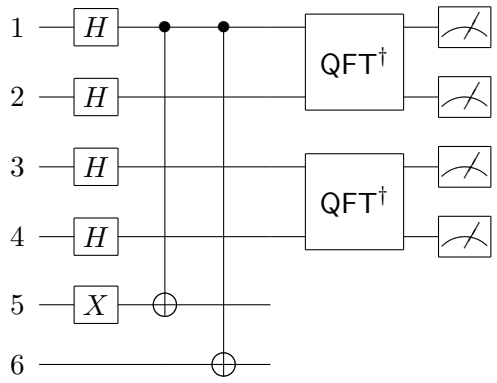

図 5.30: [AST+20] の 6 量子ビットを用いた離散対数問題の量子回路

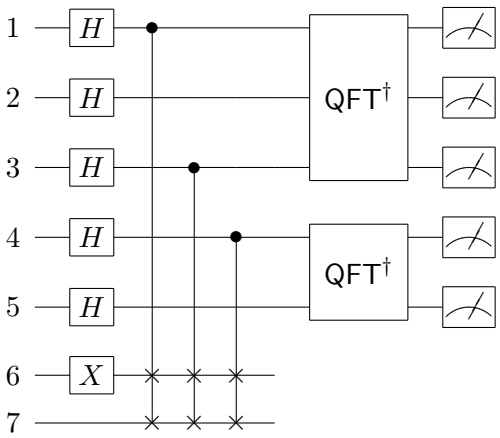

図 5.31: [AST+20] の 7 量子ビットを用いた離散対数問題の量子回路

第6章　実装の見積もり

第 5 章で確認したように，これまでいくつかの素因数分解実験が行われてきたが，いずれも小さな合成数 $N = 15, 21$ のときしか成功しておらず，計算する位数の情報を利用した実装などもあり，一般の合成数の素因数分解にはほど遠い．一般の素因数分解を行うためには，ゲート演算に誤りが生じることを考慮する必要があり，さらに，大きなパラメータの計算を行うためには非効率的な量子誤り訂正を行う必要がある．そのため，第 5 章で見たように量子誤り訂正を行わずに 35 の素因数分解や $2^x = 2 \mod 3$ の離散対数問題実験を失敗している現状では，暗号で用いるパラメータの問題を解くのは困難であると考えられる．だが，今後どのようなスピードで量子コンピュータが進化していくのかを予想するのは困難である．そのため本章では，どれだけの性能の量子コンピュータが完成すれば素因数分解や離散対数問題を解くことが可能になるのかの見積もりを行なった結果をまとめる．

6.1　実装の物理的困難性

量子コンピュータは，現状大規模な実験をするほどの性能は持ち合わせていない．ここでその理由のうちいくつかを簡単にまとめる．まず，大きな数の演算を行うためにはそのサイズに応じた量子ビット数が必要となるが，現在の量子コンピュータでは暗号で用いるパラメータで必要となるような量子ビット数を実現していない．そのため，仮に他の制約を無視したとしてもまだ暗号の攻撃に用いることは原理的にできない．次に，量子コンピュータは通常の古典コンピュータと比べるとゲート演算時の誤り率が高い．そのため，古典コンピュータと同様に誤り訂正を行うことで計算を実現するのだが，量子誤り訂正は古典誤り訂正と比べて効率が悪く，誤り訂正を行いながら計算するためには多くの量子ビットを必要とする．同様に，測定の際にも誤りが生じる場合がある．また，量子状態は長い時間維持し続けることが困難であり，古典コンピュータとは異なり早い時間で量子状態が崩れてしまう．この時間をコヒーレンス時間と呼ぶ．これらのデメリットは個別に考えてもあまり意味がない．例えば，量子ビットを増やせば増やすほど量子コンピュータは良くなるといわけではなく，量子ビットが増えれば増えるほど量子状態を維持するのが困難になり，ゲート演算の誤り率が増え，コヒーレンス時間が小さくなる．また，なるべく長く量子状態を維持しコヒーレンス時間を増やすためにはな

表 6.1: 最大演算回数の見積もり

	コヒーレンス時間（秒）	ゲート演算時間（秒）	最大演算回数
核スピン	10^{-2} – 10^{8}	10^{-3} – 10^{-6}	10^{5} – 10^{14}
電子スピン	10^{-3}	10^{-7}	10^{4}
捕捉イオン	10^{-1}	10^{-14}	10^{13}
電子 (Au)	10^{-8}	10^{-14}	10^{6}
電子 (GaAs)	10^{-10}	10^{-13}	10^{3}
量子ドット	10^{-6}	10^{-9}	10^{3}
光共振器	10^{-5}	10^{-14}	10^{9}
マイクロ波空洞	10^{0}	10^{-4}	10^{4}

るべく系が外界から遮断されている必要があるが，そのためには外界からの操作がより困難になるため，ゲート演算や測定での誤り率が増大する．このように，量子コンピュータの特性を一つの指標で評価することは難しく，様々な要素が複合的に絡み合っている．そのため，この程度の性能の量子コンピュータがあれば暗号を攻撃するのに十分であるという統一的な指標を作ることは難しいと思われる．

この小節の最後に，既存の量子コンピュータの性能について簡単にまとめる．表 6.1 は，[NC11] の Figure 7.1 を参考に，各量子コンピュータの実現法に対してコヒーレンス時間とゲート演算の時間，そしてそれらから計算される最大可能演算回数についてまとめたものである．Kunihiro [Kun05] は，1024 ビット合成数を素因数分解するためには 3074 量子ビットを用いて 2.90×10^{11} 回のゲート演算が必要だと見積もり，そのため，表 6.1（[Kun05] の Table 3）よりこの実験を実現しうるのは核スピンと捕捉イオンだけだとしている．ただし，前述の通り量子コンピュータの性能は複合的に絡み合い，[NC11] らによる表 6.1 は量子ビット数などを考慮に入れておらず，Kunihiro [Kun05] は 1024 ビット合成数を素因数分解可能であると結論でけているわけではないことに注意されたい．むしろ，コヒーレンス時間とゲート演算時間という二つの指標だけで考えても暗号で用いるパラメータの問題を解くのが難しいことがわかる．また，[KSB+20] の Table 1 と [BCMS19] の TABLE I を参考に，超伝導量子ビットと捕捉イオン量子ビットを用いた際のゲート演算の性能をそれぞれ表 6.2 と表 6.3 にまとめている．表 6.2 では 2 量子ビットゲート，表 6.3 では 1 量子ビットゲートと 2 量子ビットゲートに対する演算の成功確率と計算時間をまとめている．捕捉イオン量子ビットは，表 6.1 においても多くの演算回数が可能であり，表 6.2 と表 6.3 の 2 量子ビットゲートの行を比較しても同程度の成功確率

表 6.2: 超伝導量子ビットの 2 量子ビットゲート演算性能

	成功確率 (%)	計算時間 (ナノ秒)
[BKM+14]	99.4	40
[KSG+20]	99.7	60
[CCG+11]	90	31
[SMCG16]	99.1	160
[PGM+12]	86	800
[CGC+13]	87.2	510
[CNR+14]	99.0	30
[PMS+16]	98.5	413
[MFM+16]	98.2	183
[HPS+19]	99.2	176
[RGR+18]	∼99	190
[CBW+18]	79	0.0046

で高速に演算できることがわかっている．ただし，多くの文献で捕捉イオン量子ビットを用いた際の量子ビット数の大規模化は困難であることが指摘されていることに注意されたい．

6.2 素因数分解と ECDLP 計算のリソース評価

この章で，素因数分解と ECDLP 計算に関する最先端のリソース評価をまとめる．

表 6.4 は，[GE19] の TABLE I に倣い Shor のアルゴリズムによる素因数分解と ECDLP 計算の漸近的リソース評価をまとめたものである．本報告書で [HJN+20, BBvHL20] を追加したが，これらの論文には測定の深さに関して言及がなかったため本報告書では省略している．表 6.4 における論理量子ビット数は，Shor のアルゴリズムを実行する際に必要な量子ビット数を表している．ただし，ここでは量子誤り訂正などを考慮せずに Shor のアルゴリズムを実行する場合に必要な量子ビット数を記載している．測定の深さは，各測定において一度に測定する長さの最大値を表しており，アルゴリズムの再実行は考慮に入れていない．また，Toffoli ゲート数もアルゴリズムの再実行は考慮に入れていない．

[GE19] は現在最も効率的な素因数分解法を提案しており，特に必要な Toffoli ゲート数を大幅に省

表 6.3: 捕捉イオンビットのゲート演算性能

1 量子ビットゲート	成功確率 (%)	計算時間 (マイクロ秒)
[BXN+17]	99.995	5
[BHL+16]	99.993	7.5
[GTL+16]	99.996	2
[CMQ+10]	99	0.00005
[KGA+11]	99.9	8
[HAB+14]	99.9999	12
[OWC+11]		0.0186
2 量子ビットゲート	成功確率 (%)	計算時間 (マイクロ秒)
[EWP+19]	99.6	
[BKRB08]	99.3	50
[GTL+16]	99.91	30
[BHL+16]	99.9	100
[SBT+18]	99.8	1.6
[SBT+18]	60	0.5
[HSA+16]	99.7	3250
[WRW+16]	98.5	2700
[BSH+15]	99.8	27.4
[TGL+15]	97.9	35

表 6.4: Shor のアルゴリズムの漸近的リソース評価

	論理量子ビット数	測定の深さ	Toffoli ゲート数
素因数分解			
[VBE96]	$7n+1$	$80n^3+O(n^2)$	$80n^3+O(n^2)$
[Zal98]	$3n+O(1)$	$12n^3+O(n)$	$12n^3+O(n^2)$
[Zal98]	$5n+O(1)$	$600n^2+O(n)$	$52n^3+O(n^2)$
[Zal98]	$\approx 96n$	$\approx 2^{17}n^{1.2}$	$\approx 2^{17}n^2$
[Bea03]	$2n+3$	$144n^3\lg n+O(n^2\lg n)$	$576n^3\lg^2 n+O(n^3\lg n)$
[FMMC12]	$3n+O(1)$	$40n^3+O(n^2)$	$40n^3+O(n^2)$
[HRS17]	$2n+2$	$52n^3+O(n^2)$	$64n^3\lg n+O(n^3)$
[GE19]	$3n+0.002n\lg n$	$500n^2+n^2\log n$	$0.3n^3+0.0005n^3\lg n$
ECDLP			
[RNSL17]	$9n+O(\lg n)$	$448n^3\lg n+4090n^3$	$448n^3\lg n+4090n^3$
[HJN+20]	$8n+O(\lg n)$		$436n^3$
[BBvHL20]	$7n+O(\lg n)$		$48n^3+352n^2\log n+O(n^2)$

表 6.5: Toffoli ゲート数評価（10 億個）

素因数分解	$n = 1024$	$n = 2048$	$n = 3072$
[VBE96]	86	690	2300
[Zal98]	13	100	350
[Zal98]	56	450	1500
[Zal98]	140	550	1200
[Bea03]	62000	600000	2200000
[FMMC12]	43	340	1200
[HRS17]	580	5200	19000
[GE19]	0.4	2.7	9.9
ECDLP	$n = 160$	$n = 224$	$n = 256$
[RNSL17]	30	84	130
[HJN+20]	1.8	4.9	7.3
[BBvHL20]	0.3	0.7	1.0

略している．暗号で用いるパラメータのもとでの Toffoli ゲート数を [GE19] の TABLE I に倣い表 6.5 にまとめる．ただし，素因数分解と ECDLP で同等の古典安全性を持つように記載している．これまで同じ古典安全性を持つパラメータでは，素因数分解と ECDLP 計算において量子アルゴリズムによるリソースは前者がずっと効率的であると考えられていたが，[GE19] の結果ではこれまでの素因数分解アルゴリズムを大幅に効率化しており，ECDLP との差が大きく縮まっていることがわかる．

表 6.6 では，[GE19] の TABLE I に倣い素因数分解と ECDLP 計算を実行した場合に必要な物理量子ビット数と計算時間の積をまとめている．ここでは，表 6.4 とは異なりアルゴリズムが失敗した場合の再実行も考慮に入れたものである．これにより，[GE19] によって量子アルゴリズムによる素因数分解に必要なリソースが大幅に減少したことがわかる．

表 6.7 で，[GE19] の TABLE II に倣いこれまでに提案された 2048 ビット合成数を素因数分解するための量子リソースをまとめる．ゲートエラー率は，ゲート演算における計算失敗確率を表す．符号測定時間は，表面符号の一度の測定時間を表す．反応時間は，測定，誤り訂正の後に次の測定でどの基底を用いるのかに要する時間を表す．物理的結合は，各量子ビットが相互作用できる構造を表している．一般に，格子は超伝導量子ビットを，任意は捕捉イオン量子ビットを用いたものである．

表 6.6: 物理量子ビット数（100 万個）と計算時間（日）の積

素因数分解	$n = 1024$	$n = 2048$	$n = 3072$
[VBE96]	240	4100	23000
[Zal98]	16	250	1400
[Zal98]	16	160	540
[Zal98]	62	260	710
[Bea03]	32000	380000	1700000
[FMMC12]	53	850	4600
[HRS17]	230	2800	13000
[GE19]	0.5	5.9	21
ECDLP	$n = 160$	$n = 224$	$n = 256$
[RNSL17]	13	52	83

表 6.7: 2048 ビット合成数を素因数分解する際の Shor のアルゴリズムのリソース評価

	ゲートエラー率	符号測定時間（マイクロ秒）	反応時間（マイクロ秒）	物理的結合
[FMMC12]	0.1%	1	0.1	格子
[OC17]	0.1%	10	1	任意
[GM19]	0.1%	0.2	0.1	格子
[GE19]	0.1%	1	10	格子

	物理量子ビット数（100 万個）	平均計算時間（日）
[FMMC12]	1000	1.1
[OC17]	230	3.7
[GM19]	170	1
[GE19]	20	0.31

6.3 [GM19] によるトレードオフ評価

Gheorghiu と Mosca [GM19] は，格子状の結合に対して表面符号を用いた実装技法 [HFDM12, FG19, Lit19] と [Bea03, BBC+95, CDKM04, RNSL17] の Shor のアルゴリズムを用いたときの素因数分解と ECDLP 計算のリソース評価を行なった．Gheorghiu と Mosca は，y を 2 を底とする量子ビット数，x を 2 を低とする計算時間（秒）として 3 次関数

$$y(x) = \alpha x^3 + \beta x^2 + \gamma x + \delta \tag{6.1}$$

における係数 $\alpha, \beta, \gamma, \delta$ を計算した．これによって，Shor のアルゴリズムにおける時間-メモリトレードオフを確認することができる．特に，

$$y(\log_2(24 \times 3600)) \approx y(16.3987)$$

を計算することで 1 日でアルゴリズムを実行するための量子ビット数を得ることができる．そのため，表 6.7 では 1 回あたりの実行での計算時間が 1 日となるときの量子ビット数が記載されている．表 6.7 に記載されているように，表面符号の測定時間は 0.2 マイクロ秒であるが，この数値は [FMMC12] による．また，Gheorghiu と Mosca は，ゲート演算の誤り率が $p = 10^{-3}$ と $p = 10^{-5}$ の二つの場合について評価を行なっており，誤り率に関するトレードオフも確認することができる．

Gheorghiu と Mosca は，1024, 2048, 3072, 4096, 7680, 15360 ビット合成数の素因数分解と楕円曲線 NIST P-160, NIST P-192, NIST P-224, NIST P-256, NIST P-384, NIST P-521 にける離散対数問題に対してこの評価を行なった．表 6.8 にそれぞれの式 (6.1) の係数をまとめる．また，図 6.2–6.12 にそれぞれの合成数や楕円曲線に対するリソース評価の図を記す．いずれも横軸が 2 を低とする計算時間の対数，縦軸が 2 を低とする量子ビット数の対数となっており，誤り率が $p = 10^{-3}$ と $p = 10^{-5}$ の曲線を引いている．

6.4 [GE19] による素因数分解の効率化技法

第 6.2 で確認したように，Gidney と Ekerå [GE19] の素因数分解アルゴリズムは暗号で用いるパラメータにおいては既存の最も効率的な手法であると言える．Gidney と Ekerå は，既存の様々な効率化技法を組み合わせることでこの改良を達成しており，本節でそれらの一部を説明する．ただし，Gidney と Ekerå は半古典フーリエ変換も用いているが，第 5.2.2 節で説明したのでここでは省略する．

表 6.8: [GM19] による式 (6.1) の係数

	古典安全性（ビット）	誤り率	α	β	γ	δ
RSA-1024	80	10^{-3}	0.0022	−0.0639	−0.4071	38.999
RSA-1024	80	10^{-5}	0.0021	−0.05	−0.6216	35.405
RSA-2048	112	10^{-3}	0.0015	−0.0437	−0.6545	43.227
RSA-2048	112	10^{-5}	0.0014	−0.0316	−0.8545	39.558
RSA-3072	128	10^{-3}	0.001	−0.0163	−1.1477	48.049
RSA-3072	128	10^{-5}	0.0009	−0.0107	−1.2235	43.576
RSA-4096	156	10^{-3}	0.0006	0.0019	−1.5324	52.111
RSA-4096	156	10^{-5}	0.0001	0.0357	−2.1075	50.283
RSA-7680	192	10^{-3}	−0.0003	0.0647	−3.2994	74.195
RSA-7680	192	10^{-5}	0.0019	−0.0962	0.6053	40.352
RSA-15360	256	10^{-3}	−0.0013	0.1585	−6.192	106.79
RSA-15360	256	10^{-5}	0.0012	−0.0532	−0.3513	50.929
NIST P-160	80	10^{-3}	0.0024	−0.0737	−0.2862	38.04
NIST P-160	80	10^{-5}	0.0023	−0.0579	−0.5353	34.598
NIST P-192	96	10^{-3}	0.0023	−0.0719	−0.279	38.774
NIST P-192	96	10^{-5}	0.0022	−0.0577	−0.5145	35.305
NIST P-224	112	10^{-3}	0.0022	−0.0712	−0.626	39.301
NIST P-224	112	10^{-5}	0.0021	−0.0562	−0.5168	35.954
NIST P-256	128	10^{-3}	0.0021	−0.0698	−0.2624	39.826
NIST P-256	128	10^{-5}	0.002	−0.054	−0.5327	36.584
NIST P-384	192	10^{-3}	0.0017	−0.0567	−0.4258	42.49
NIST P-384	192	10^{-5}	0.0016	−0.043	−0.6715	39.128
NIST P-521	256	10^{-3}	0.0013	−0.0357	−0.7883	45.97
NIST P-521	256	10^{-5}	0.0012	−0.0294	−0.8999	41.829

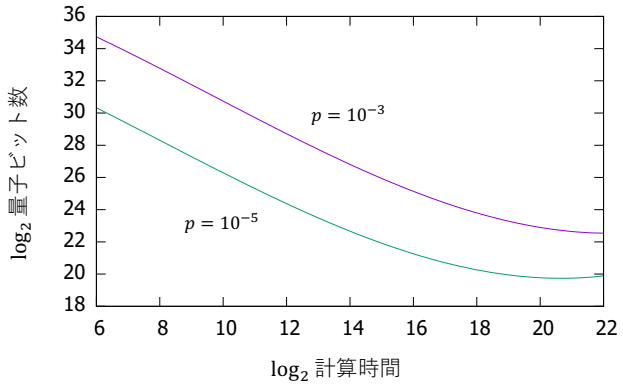

図 6.1: RSA-1024

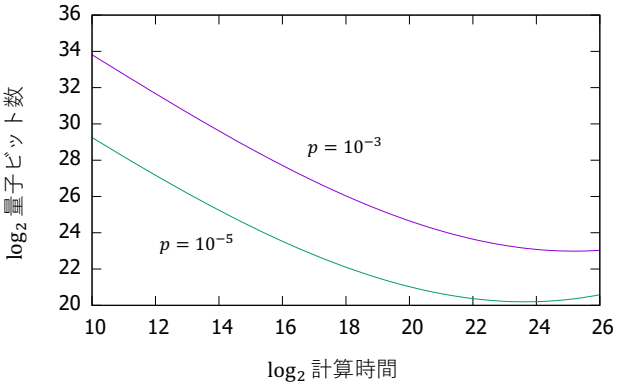

図 6.2: RSA-2048

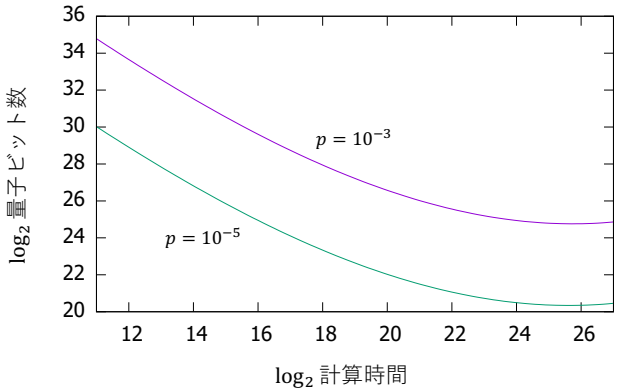

図 6.3: RSA-3072

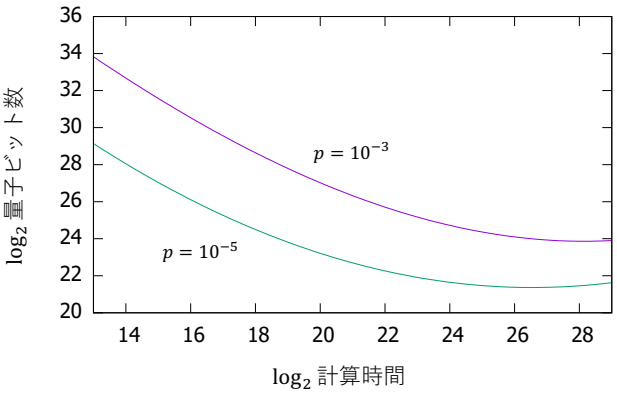

図 6.4: RSA-4096

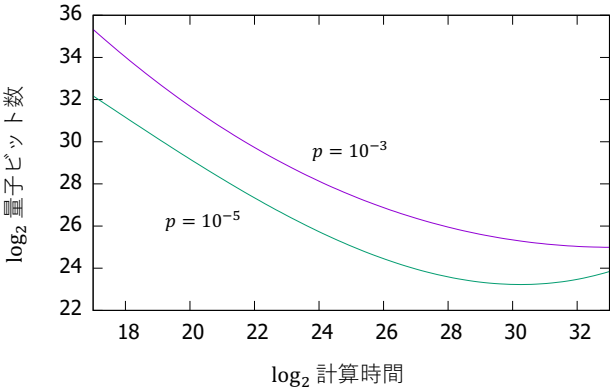

図 6.5: RSA-7680

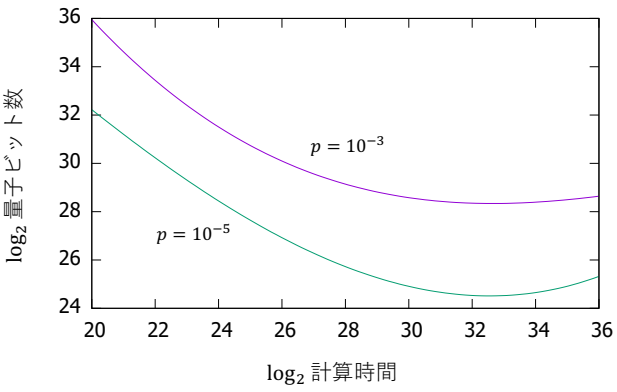

図 6.6: RSA-15630

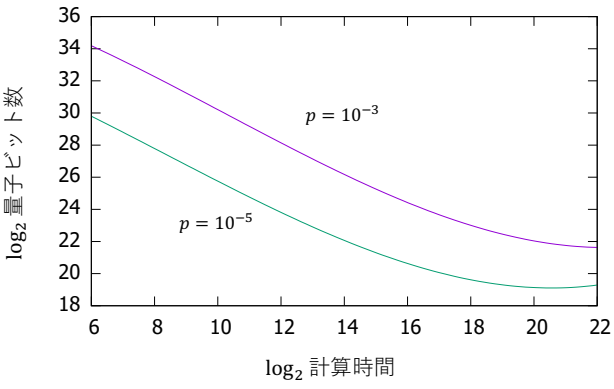

図 6.7: P-160

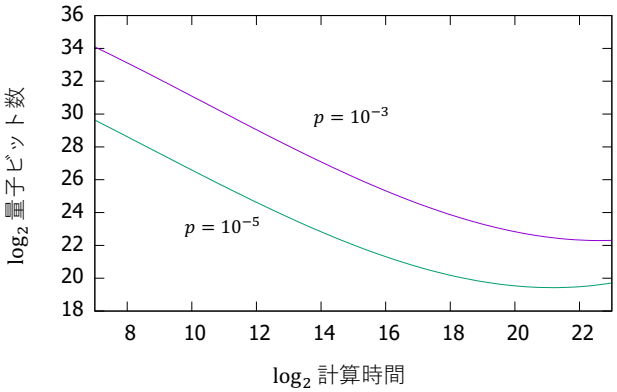

図 6.8: P-192

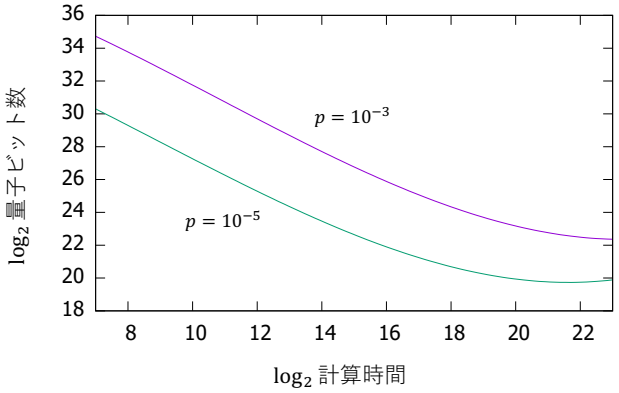

図 6.9: P-224

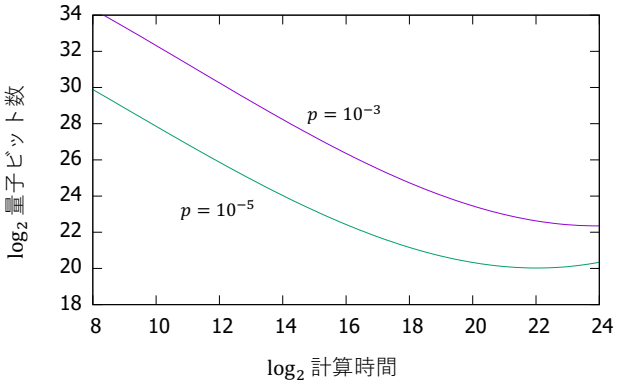

図 6.10: P-256

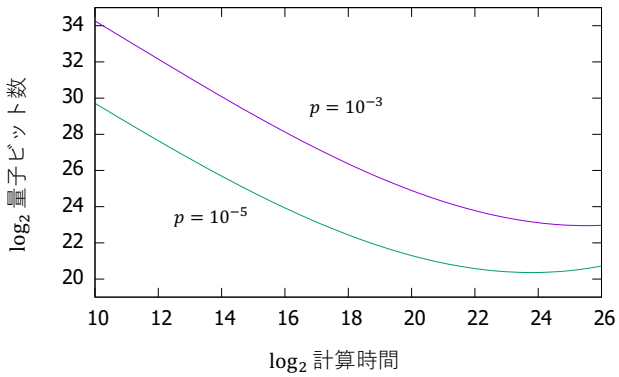

図 6.11: P-384

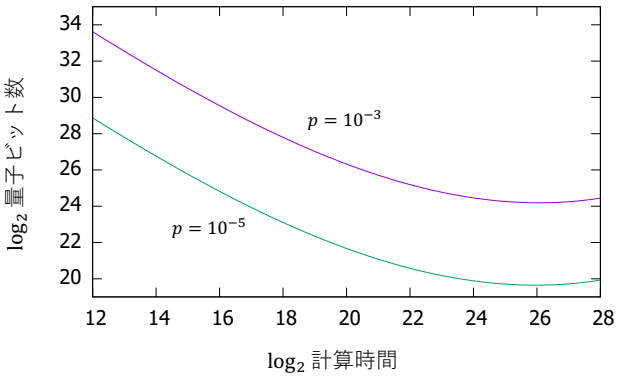

図 6.12: P-521

6.4.1 離散対数問題を解くことによる素因数分解

Ekerå と Håstad [Eke16, EH17, Eke20] は，Shor のアルゴリズムではなく離散対数問題を解くことによる素因数分解アルゴリズムを提案した．より正確に言えば，小さな離散対数を求めることによって素因数分解できることを示した．n ビットの合成数 $N = pq$ を素因数分解するために，まず $\mathbb{Z}_N^*$ の元 g を一様ランダムに選び，$y = g^{N+1}$ を古典的に計算する．乗法群 $\mathbb{Z}_N^*$ の位数は $(p-1)(q-1)$ である．ここで，N が十分大きいときには g の位数 r は高確率で $r > p+q$ を満たし，$(p-1)(q-1)$ の約数である．そのため，$d \coloneqq \log_g y \mod r$ とすると，

$$d = pq + 1 = p + q \mod r$$

を満たす．離散対数アルゴリズムによって $d = p + q$ が計算できれば，$N = pq$ より二次方程式 $x^2 - dx + N = 0$ を解くことで N を素因数分解することができる．m を $p + q < 2^m$ を満たす整数とする．関数

$$f(e_1, e_2) = g^{e_1} y^{-e_2}$$

の周期を計算することで上記の離散対数問題を解くと，e_1 と e_2 をそれぞれ $2m$ ビットと m ビットとし，全体として第 1 レジスタに必要な量子ビット数は

$$n_e \coloneqq 3m = 1.5n + O(1)$$

となり，これは Shor のアルゴリズムの $2n$ より小さくなっていることがわかる．また，周期計算アルゴリズム実行後には古典的に格子アルゴリズムを適用することで d を計算することができることがわかっている [Eke20].

6.4.2 制御冪乗演算の分解

位相計算アルゴリズムの実装において，[Bea03, Gid17, HRS17, VBE96, Zal98, Zal06] と同様に制御冪乗剰余演算を制御乗法剰余演算の組み合わせで記述する．これは，第 4.3.2 小節で説明したように m ビット整数 x の冪乗演算 a^x を計算するために，$x = x_0 + 2x_1 + \cdots + 2^{m-1}x_{m-1}$ に対して $i \in \{i_1, \ldots, i_{m'}\} \subseteq \{0, 1, \ldots, m-1\}$ が常に $x_i = 1$ を満たし，$i \in \{0, 1, \ldots, m-1\} \setminus \{i_1, \ldots, i_{m'}\}$ が常に $x_i = 0$ を満たすとき，

$$a^x = a^{x_0 + 2x_1 + \cdots + 2^{m-1}x_{m-1}}$$

$$
\begin{aligned}
&= a^{x_0} \cdot a^{2x_1} \cdot \cdots \cdot a^{2^{m-1}x_{m-1}} \\
&= a^{2^{i_1}} \cdots a^{2^{i_{m'}}}
\end{aligned}
$$

を計算する．ここで，$a^2 \mod N, \ldots, a^{2^{m-1}} \mod N$ は事前に古典的に計算しておく．

次に，ある a に対する乗法剰余演算 $|x \mod N\rangle \to |a \cdot x \mod N\rangle$ を考える．この演算は効率的に行うことができない．そのため，$|0\rangle$ と初期化された n ビットの補助レジスタを用いて $|x\rangle|0\rangle \to |ax\rangle|0\rangle$ と計算を行う．まず，第 1 レジスタの a 倍を第 2 レジスタに加えて

$$
|x\rangle|0\rangle \to |x\rangle|ax\rangle
$$

を計算する．次に，$a^{-1} \mod N$ を事前に古典的に計算しておくことで，第 2 レジスタの値の $a^{-1} \mod N$ 倍を第 1 レジスタから引くことで

$$
\begin{aligned}
|x\rangle|ax\rangle &\to |x - a^{-1} \cdot (ax)\rangle|ax\rangle \\
&= |0\rangle|ax\rangle
\end{aligned}
$$

を計算する．最後に，二つのレジスタをスワップして計算結果 $|ax\rangle|0\rangle$ を得る．

さらに，上記の計算を行うためには，整数倍を足す $|x\rangle|y\rangle \to |x\rangle|ax + y\rangle$ という演算が必要である．そのためには，最初に冪乗演算を乗算の組み合わせに分解したときと同様，乗算を加算の組み合わせに分解する．つまり，$x = x_0 + 2x_1 + \cdots + 2^{m-1}x_{m-1}$ に対して $i \in \{i_1, \ldots, i_{m'}\} \subseteq \{0, 1, \ldots, m-1\}$ が常に $x_i = 1$ を満たし，$i \in \{0, 1, \ldots, m-1\} \setminus \{i_1, \ldots, i_{m'}\}$ が常に $x_i = 0$ を満たすとき，

$$
\begin{aligned}
ax &= a \cdot (x_0 + 2x_1 + \cdots + 2^{m-1}x_{m-1}) \\
&= ax_0 + 2ax_1 + \cdots + 2^{m-1}ax_{m-1} \\
&= 2^{i_1}a + \cdots + 2^{i_{m'}}a
\end{aligned}
$$

を計算する．ここで，$2a \mod N, \ldots, 2^{m-1}a \mod N$ は事前に古典的に計算しておく．よって，制御加法剰余計算は加算と比較の組み合わせで計算することができる．[VBE96] によると，このためには 5 回の加法で十分であり，[CDKM04] によると加算は $O(1)$ 個の補助ビットと $2n$ 個の Toffoli ゲートで実行可能である．

これらをまとめると，第 6.4.1 節のアルゴリズムを実行するためには $n_e \cdot 2n \cdot 5 \cdot 2n = 20n_e n^2$ 個の Toffoli ゲートが必要である．

6.4.3　剰余類表現

これまで，$k \mod N$ は量子状態としては $|k\rangle$ として表現されてきて，Shor のアルゴリズムでは冪乗剰余演算が必要であった．Zalka [Zal06] は，剰余類表現 (coset representation) を用いて，冪乗剰余演算を冪乗演算に置き換える手法を提案した．剰余類表現においては，あるパディングのための補助ビット数を表すパラメータ c_{pad} のもとで，$k \mod N$ を

$$\frac{1}{\sqrt{2^{c_{pad}}}} \sum_{j=0}^{c_{\mathsf{pad}}-1} |jN + k\rangle$$

という量子状態で記述する．剰余類表現は，剰余演算を厳密にではなく近似的に行うことに注意されたい．この表現法によって $10n$ 個の Toffoli ゲートが必要だった冪乗剰余演算を $4n$ 個の Toffoli ゲートを用いる冪乗演算に置き換えることができ [CDKM04, Gid18]，全体として第 6.4.2 節で述べた $20n_e n^2$ 個 Toffoli ゲートの数を $8n_e n^2$ まで減らすことができる．

6.4.4　ウィンドウ法

Gidney [Gid19a] は，ウィンドウ法を用いることで冪乗計算における Toffoli ゲートの数を減らした．乗算を行う際に，いくつかの制御加算を事前計算結果の加算に置き換えた．この事前計算結果は，テーブルを参照することで得られる．よって，まとめられた複数の制御ビットは，古典的に計算された事前計算テーブルの番地を表す．同様に，冪乗演算を行う際のいくつかの制御乗算を，それらの制御ビットによって表される番地の事前計算テーブルの参照に置き換えた．

それぞれのウィンドウ幅を c_{mul} と c_{exp} と書くことにする．このとき，n_e 回の制御乗算は n_e/c_{exp} 回の乗算に置き換えられ，各乗算における $2n$ 回の制御加算は $2n/c_{\mathsf{mul}}$ 回の加算に置き換えられる．ただし，一度の加算のために $c_{\mathsf{exp}} + c_{\mathsf{mul}}$ 個の事前計算結果が必要になる．

Cuccaro ら [CDKM04] の結果によると，各 n ビット加算には $2n$ 個の Toffoli ゲートと深さ $2n$ の測定が必要である．Babbush ら [BGB+18] の結果によると，各テーブル参照には $2^{c_{\mathsf{exp}}+c_{\mathsf{mul}}}$ 個の Toffoli ゲートと深さ $2^{c_{\mathsf{exp}}+c_{\mathsf{mul}}}$ の測定，さらに補助ビットが $O(c_{\mathsf{exp}} + c_{\mathsf{mul}})$ 必要である．さらにその後，この操作は [BGM+19] によると測定型量子計算によって $2\sqrt{2^{c_{\mathsf{exp}}+c_{\mathsf{mul}}}}$ 個の Toffoli ゲートと深さ $2\sqrt{2^{c_{\mathsf{exp}}+c_{\mathsf{mul}}}}$ の測定，さらに補助ビットが $\max(0, \sqrt{2^{c_{\mathsf{exp}}+c_{\mathsf{mul}}}} - n)$ 必要である．ただし，この補助ビットは c_{exp} と c_{mul} を n に比べて小さな定数とすることで常に 0 にすることができる．

まとめると，ウィンドウ法を用いることで，小さな項を無視すると Toffoli ゲートの数を $\frac{2n_e n}{c_{\mathsf{exp}} c_{\mathsf{mul}}}(2n + 2^{c_{\mathsf{exp}}+c_{\mathsf{mul}}})$ まで減らすことができる．$c_{\mathsf{exp}} = c_{\mathsf{mul}} = \frac{1}{2}\lg n$ と置くことで，この値は $\frac{24n_e^2 n}{\lg^2 n}$ となる．空間

計算量としては，乗算結果を記録するために $n + O(\lg n)$ ビットのレジスタ，乗算の途中結果を記録するために $n + O(\lg n)$ ビットのレジスタ，参照した事前計算結果を出力するための $n + O(\lg n)$ ビットのレジスタ，そして半古典フーリエ変換を行うために $\lg n$ ビットのレジスタを使用し，全体の量子ビット数は $3n + O(\lg n)$ となる．

6.4.5 忘却繰り上げ

一般に，各種計算の際の繰り上げ操作は複数の量子ビットを順に処理していく必要がある．この操作を効率化するために，忘却繰り上げ [Gid19b] を用いる．これによって，大きな数の加算を行う際に各加算を要素ごとに並列で行うことが可能になる．剰余類表現と同様，忘却繰り上げは厳密ではなく近似的な加算を行うことに注意されたい．ただし，繰り上げ回路の大きさ c_{pad} の大きさを適切に設定することでこの近似による誤り確率を十分小さく抑えることができる．c_{sep} を繰り上げ回路の分割を表すパラメータとすると， [DKRS06] らによる繰り上げの効率化とは異なり，Toffoli ゲート数と追加コストは $\lceil n/c_{\mathsf{cep}} \rceil$ に線形で n には対数となる．これにより，量子ビット数と Toffoli ゲート数を少し増やすだけで並列化を可能にし，全体の計算が効率化される．

6.4.6 効率化技法を用いた際のリソース評価

これまでの効率化技法を用いたときの量子回路のリソース評価を行う．ここまで用いたパラメータとして $n, n_e, c_{\mathsf{mul}}, c_{\mathsf{exp}}, c_{\mathsf{sep}}, c_{\mathsf{pad}}$ があった．Gidney と Ekerå は，$c_{\mathsf{mul}} = c_{\mathsf{exp}} = 5, c_{\mathsf{sep}} = 1024, c_{\mathsf{pad}} = 2\lg n + \lg n_e + 10 \approx 3\lg n + 10$ のときに必ずしも最適ではないが十分効率的になるとした．これまでように，n_e ビットの冪乗演算は c_{exp} の大きさの n_e 回の乗算に置き換えられ，各乗算において 2 回の乗算と加算を行い，この操作は n 個の主レジスタに応じて行い，剰余類表現のパディングのための $O(\lg n)$ 量子ビットと忘却繰り上げのための n/c_{cep} 量子ビットを要する．さらに，これらの加算はウィンドウ法を用いて c_{mul} の大きさのテーブル参照による加算を行う．そのため，前述のパラメータ設定によるとテーブル参照による加算の総数は

$$\frac{2nn_e}{c_{\mathsf{exp}}c_{\mathsf{mul}}} \cdot \frac{c_{\mathsf{sep}} + 1}{c_{\mathsf{mul}}} + O\left(\frac{n_e \lg n}{c_{\mathsf{exp}}c_{\mathsf{mul}}}\right) \to \frac{2nn_e}{25} \cdot \frac{1025}{1024} + O\left(n_e \lg n\right) \approx 0.1n_e n$$

となる．このテーブル参照による加算がほとんどの計算を占めるため，Toffoli ゲートの総数は

$$0.1n_e n \cdot \left(2n + c_{\mathsf{pad}} \frac{n}{c_{\mathsf{sep}}} + 2^{c_{\mathsf{exp}} + c_{\mathsf{pad}}}\right) \approx 0.2n_e n^2 + 0.0003n_e n^2 \lg n$$

となる．同様に，測定の深さは

$$0.1n_e n \cdot \left(2c_{\mathsf{sep}} + 2c_{\mathsf{pad}} + 2^{c_{\mathsf{exp}}+c_{\mathsf{pad}}}\right) \approx 300n_e n + 0.5n_e n \lg n$$

となる．$n_e = 1.5n$ として，表 6.4 の値を得る．さらに，エラー確率の解析や量子誤り訂正を適用することで表 6.7 の結果を得る．

表 6.9: [GE19] によって n ビット合成数を素因数分解する際の Shor のアルゴリズムのリソース評価

ビット数 n	物理量子ビット数（100 万）	平均計算時間（時間）	失敗確率
1024	9.7	1.3	6%
2048	20	5.1	31%
3072	38	12	9%
4096	55	22	5%
8192	140	86	5%
12288	200	200	12%
16384	270	350	24%

表 6.9 は，Gidney と Ekerå が評価した様々な大きさの合成数の素因数分解のためのリソース評価である．ここで，2048 ビット合成数を素因数分解するための計算時間が表 6.7 とは異なっているが，各ビット数に対してパラメータを取り直すことで高速化している．

6.4.7 素体上の離散対数問題

これまで述べた Gidney と Ekerå の効率化技法は，逆元計算が必要になる ECDLP には適用できないが素体上の離散対数問題においても効率化が可能である．$\mathbb{Z}_N^*$ における位数 r の元 g と h が与えられたときに離散対数 $d = \log_g h$ を計算する離散対数問題を考える．ここで，r は $N-1$ を割り切るので，正整数 $k \geq 1$ を用いて $N = 2rk + 1$ と書くことにし，z を一般数体篩法 [Gor93] による古典安全性，n_d と n_r をそれぞれ d と r のビット長とする．この安全性を達成するために，$n_d, n_r \geq 2z$ とする．ここでは，$n_d = n_r = 2z$ とする Scnorr 群と $n_r = n - 1$ とする安全素数群を考える．また，安全素数群において $n_d = 2z$ となるとき，離散対数は小さいと呼ぶことにする．量子アルゴリズムによってこの素体上の離散対数問題を解くとき，安全素数群において小さな離散対数問題を解く最も

効率的なアルゴリズムは Ekerå と Håstad のアルゴリズム [EH17, Eke20] である．それ以外の場合，Ekerå のアルゴリズム [Eke18] は効率的である．また，群の位数が未知のときには Shor のアルゴリズム [Sho94, Sho97] が効率的となり， [Eke16, Eke19] に詳細な解析が書かれている．これらのアルゴリズムに前述の Gidney と Ekerå の効率化技法を組み合わせたときのそれぞれのリソース評価を表 6.10–6.14 に示す．

表 6.10: [EH17, Eke20] のアルゴリズムによる安全素数群の小さな離散対数計算のリソース評価

ビット数 n	物理量子ビット数（100 万）	平均計算時間（時間）	失敗確率
1024	9.2	0.4	10%
2048	20	1.2	9%
3072	29	2.0	18%
4096	51	3.1	4%
8192	110	8.3	5%
12288	170	15	9%
16384	220	23	17%

表 6.11: [Eke18] のアルゴリズムによる Schnorr 群の離散対数計算のリソース評価

ビット数 n	物理量子ビット数（100 万）	平均計算時間（時間）	失敗確率
1024	9.2	0.4	10%
2048	20	1.2	9%
3072	29	2.0	18%
4096	51	3.1	4%
8192	110	8.3	5%
12288	170	15	9%
16384	220	23	17%

表 6.12: [Eke18] のアルゴリズムによる安全素数群の離散対数計算のリソース評価

ビット数 n	物理量子ビット数（100 万）	平均計算時間（時間）	失敗確率
1024	9.7	2.7	10%
2048	26	11	6%
3072	41	24	5%
4096	55	43	9%
8192	140	180	8%
12288	200	390	21%
16384	320	700	16%

表 6.13: [Sho94, Sho97] のアルゴリズムによる Schnorr 群の離散対数計算のリソース評価

ビット数 n	物理量子ビット数（100 万）	平均計算時間（時間）	失敗確率
1024	9.2	0.3	8%
2048	20	0.8	6%
3072	29	1.3	13%
4096	39	2.1	25%
8192	110	5.5	11%
12288	170	10	8%
16384	220	16	12%

表 6.14: [Sho94, Sho97] のアルゴリズムによる安全素数群の離散対数計算のリソース評価

ビット数 n	物理量子ビット数（100 万）	平均計算時間（時間）	失敗確率
1024	9.7	1.8	8%
2048	26	7.0	5%
3072	38	16	12%
4096	55	29	8%
8192	140	120	6%
12288	200	260	15%
16384	320	470	12%

関連図書

[ASK19] Mirko Amico, Zain H. Saleem, and Muir Kumph. Experimental study of shor's factoring algorithm using the ibm q experience. *Phys. Rev. A*, 100:012305, 2019.

[AST+20] 青野良範，Sitong Liu, 田中智樹，宇野隼平，Rodney Van Meter, 篠原直行，野島良．超伝導量子回路を用いた離散対数問題の求解実験．量子情報技術研究会 (2020-12-QIT), 2020.

[BBC+95] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Phys. Rev. A*, 52:3457–3467, 1995.

[BBvHL20] Gustavo Banegas, Daniel J. Bernstein, Iggy van Hoof, and Tanja Lange. Concrete quantum cryptanalysis of binary elliptic curves. *IACR Cryptol. ePrint Arch.*, 2020:1296, 2020.

[BCDP96] David Beckman, Amalavoyal N. Chari, Srikrishna Devabhaktuni, and John Preskill. Efficient networks for quantum factoring. *Phys. Rev. A*, 54:1034–1063, 1996.

[BCMS19] Colin D. Bruzewicz, John Chiaverini, Robert McConnell, and Jeremy M. Sage. Trapped-ion quantum computing: Progress and challenges. *Applied Physics Reviews*, 6(2):021314, 2019.

[Bea03] S. Beauregard. Circuit for shor's algorithm using $2n + 3$ qubits. *Quantum Inf. Comput.*, 3:175–185, 2003.

[BEL+16] Daniel J. Bernstein, Susanne Engels, Tanja Lange, Ruben Niederhagen, Christof Paar, Peter Schwabe, and Ralf Zimmermann. Faster discrete logarithms on fpgas. *IACR Cryptol. ePrint Arch.*, 2016:382, 2016.

[BGB+18] Ryan Babbush, Craig Gidney, Dominic W. Berry, Nathan Wiebe, Jarrod McClean, Alexandru Paler, Austin Fowler, and Hartmut Neven. Encoding electronic spectra in quantum circuits with linear t complexity. *Phys. Rev. X*, 8:041015, 2018.

[BGG+20] Fabrice Boudot, Pierrick Gaudry, Aurore Guillevic, Nadia Heninger, Emmanuel Thomé, and Paul Zimmermann. Comparing the difficulty of factorization and discrete logarithm: A 240-digit experiment. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Proceedings, Part II*, volume 12171 of *Lecture Notes in Computer Science*, pages 62–91. Springer, 2020.

[BGM+19] Dominic W. Berry, Craig Gidney, Mario Motta, Jarrod R. McClean, and Ryan Babbush. Qubitization of arbitrary basis quantum chemistry leveraging sparsity and low rank factorization. *Quantum*, 3:208, 2019.

[BHL+16] C. J. Ballance, T. P. Harty, N. M. Linke, M. A. Sepiol, and D. M. Lucas. High-fidelity quantum logic gates using trapped-ion hyperfine qubits. *Phys. Rev. Lett.*, 117:060504, 2016.

[BKM+14] R. Barends, J. Kelly, A. Megrant, A. Veitia, D. Sank, E. Jeffrey, T. C. White, J. Mutus, A. G. Fowler, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, C. Neill, P. O'Malley, P. Roushan, A. Vainsencher, J. Wenner, A. N. Korotkov, A. N. Cleland, and John M. Martinis. Superconducting quantum circuits at the surface code threshold for fault tolerance. *Nature*, 508:500–503, 2014.

[BKRB08] Jan Benhelm, Gerhard Kirchmair, Christian F. Roos, and Rainer Blatt. Towards fault-tolerant quantum computing with trapped ions. *Nature Physics*, 4:463–466, 2008.

[BSH+15] C. J. Ballance, V. M. Schäfer, J. P. Home, D. J. Szwer, S. C. Webster, D. T. C. Allcock, N. M. Linke, T. P. Harty, D. P. L. Aude Craik, D. N. Stacey, A. M. Steane, and D. M. Lucas. Hybrid quantum logic and a test of Bell's inequality using two different atomic isotopes. *Nature*, 528:384–386, 2015.

[BXN+17] A. Bermudez, X. Xu, R. Nigmatullin, J. O'Gorman, V. Negnevitsky, P. Schindler, T. Monz, U. G. Poschinger, C. Hempel, J. Home, F. Schmidt-Kaler, M. Biercuk, R. Blatt, S. Benjamin, and M. Müller. Assessing the progress of trapped-ion processors towards fault-tolerant quantum computation. *Phys. Rev. X*, 7:041061, 2017.

[CBW+18] Kevin S. Chou, Jacob Z. Blumoff, Christopher S. Wang, Philip C. Reinhold, Christopher J. Axline, Yvonne Y. Gao, L. Frunzio, M. H. Devoret, Liang Jiang, and R. J. Schoelkopf.

Deterministic teleportation of a quantum gate between two logical qubits. *Nature*, 561:368–373, 2018.

[CCG+11] Jerry M. Chow, A. D. Córcoles, Jay M. Gambetta, Chad Rigetti, B. R. Johnson, John A. Smolin, J. R. Rozen, George A. Keefe, Mary B. Rothwell, Mark B. Ketchen, and M. Steffen. Simple all-microwave entangling gate for fixed-frequency superconducting qubits. *Phys. Rev. Lett.*, 107:080502, 2011.

[CDKM04] S. A. Cuccaro. T. G. Draper, S. A. Kutin, and D. P. Moulton. A new quantum ripple-carry addition circuit. arXiv preprint quantum-ph/0601097, 2004.

[CGC+13] Jerry M Chow, Jay M Gambetta, Andrew W Cross, Seth T Merkel, Chad Rigetti, and M Steffen. Microwave-activated conditional-phase gate for superconducting qubits. *New Journal of Physics*, 15(11):115012, 2013.

[CMQ+10] W. C. Campbell, J. Mizrahi, Q. Quraishi, C. Senko, D. Hayes, D. Hucul, D. N. Matsukevich, P. Maunz, and C. Monroe. Ultrafast gates for single atomic qubits. *Phys. Rev. Lett.*, 105:090502, 2010.

[CNR+14] Yu Chen, C. Neill, P. Roushan, N. Leung, M. Fang, R. Barends, J. Kelly, B. Campbell, Z. Chen, B. Chiaro, A. Dunsworth, E. Jeffrey, A. Megrant, J. Y. Mutus, P. J. J. O'Malley, C. M. Quintana, D. Sank, A. Vainsencher, J. Wenner, T. C. White, Michael R. Geller, A. N. Cleland, and John M. Martinis. Qubit architecture with high coherence and fast tunable coupling. *Phys. Rev. Lett.*, 113:220502, 2014.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Information Theory*, 22(6):644–654, 1976.

[DKRS06] Thomas G. Draper, Samuel A. Kutin, Eric M. Rains, and Krysta M. Svore. A logarithmic-depth quantum carry-lookahead adder. *Quantum Information and Computation*, 6(4):351–369, 2006.

[DLQ+20] Zhao-Chen Duan, Jin-Peng Li, Jian Qin, Ying Yu, Yong-Heng Huo, Sven Höfling, Chao-Yang Lu, Nai-Le Liu, Kai Chen, and Jian-Wei Pan. Proof-of-principle demonstration of compiled Shor's algorithm using a quantum dot single-photon source. *Optics Express*, 28:18917–18930, 2020.

[DSBP18] Avinash Dash, Deepankar Sarmah, B. Behera, and P. Panigrahi. Exact search algorithm to factorize large biprimes and a triprime on ibm quantum computer. *arXiv: Quantum Physics*, 2018.

[EH17] Martin Ekerå and Johan Håstad. Quantum algorithms for computing short discrete logarithms and factoring RSA integers. In Tanja Lange and Tsuyoshi Takagi, editors, *Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017, Proceedings*, volume 10346 of *Lecture Notes in Computer Science*, pages 347–363. Springer, 2017.

[Eke16] Martin Ekerå. Modifying shor's algorithm to compute short discrete logarithms. *IACR Cryptol. ePrint Arch.*, 2016:1128, 2016.

[Eke18] Martin Ekerå. Quantum algorithms for computing general discrete logarithms and orders with tradeoffs. *IACR Cryptol. ePrint Arch.*, 2018:797, 2018.

[Eke19] Martin Ekerå. Revisiting shor's quantum algorithm for computing general discrete logarithms. *CoRR*, abs/1905.09084, 2019.

[Eke20] Martin Ekerå. On post-processing in the quantum algorithm for computing short discrete logarithms. *Des. Codes Cryptogr.*, 88(11):2313–2335, 2020.

[EWP+19] Alexander Erhard, Joel James Wallman, Lukas Postler, Michael Meth, Roman Stricker, Esteban Adrian Martinez, Philipp Schindler, Thomas Monz, Joseph Emerson, and Rainer Blatt. Characterizing large-scale quantum computers via cycle benchmarking. arXiv:1902.08543, 2019.

[FG19] Austin G. Fowler and Craig Gidney. Low overhead quantum computation using lattice surgery, 2019.

[FMMC12] Austin G. Fowler, Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland. Surface codes: Towards practical large-scale quantum computation. *Phys. Rev. A*, 86:032324, 2012.

[Gid17] C. Gidney. Factoring with $n + 2$ clean qubits and $n - 1$ dirty qubits. arXiv preprint quantum-ph/1706.07884, 2017.

[Gid18] Craig Gidney. Halving the cost of quantum addition. *Quantum*, 2:74, 2018.

[Gid19a] C. Gidney. Windowed quantum arithmetic. arXiv preprint arXiv:1905.07682, 2019.

[Gid19b] Craig Gidney. Approximate encoded permutations and piecewise quantum adders, 2019.

[GE19] C. Gidney and M. Ekerå. How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. arXiv preprint arXiv:1905.09749, 2019.

[GM19] Vlad Gheorghiu and Michele Mosca. Benchmarking the quantum cryptanalysis of symmetric, public-key and hash-based cryptographic schemes, 2019.

[GN93] Robert B. Griffiths and Chi-Sheng Niu. Semiclassical fourier transform for quantum computation. *Phys. Rev. Lett.*, 76(17):3228, 1993.

[Gor93] Daniel M. Gordon. Discrete logarithms in *GF(P)* using the number field sieve. *SIAM J. Discrete Math.*, 6(1):124–138, 1993.

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 212–219. ACM, 1996.

[GTL+16] J. P. Gaebler, T. R. Tan, Y. Lin, Y. Wan, R. Bowler, A. C. Keith, S. Glancy, K. Coakley, E. Knill, D. Leibfried, and D. J. Wineland. High-fidelity universal gate set for $^{9}Be^{+}$ ion qubits. *Phys. Rev. Lett.*, 117:060505, 2016.

[HAB+14] T. P. Harty, D. T. C. Allcock, C. J. Ballance, L. Guidoni, H. A. Janacek, N. M. Linke, D. N. Stacey, and D. M. Lucas. High-fidelity preparation, gates, memory, and readout of a trapped-ion quantum bit. *Phys. Rev. Lett.*, 113:220501, 2014.

[HFDM12] Clare Horsman, Austin G Fowler, Simon Devitt, and Rodney Van Meter. Surface code quantum computing by lattice surgery. *New Journal of Physics*, 14(12):123011, 2012.

[HJN+20] Thomas Häner, Samuel Jaques, Michael Naehrig, Martin Roetteler, and Mathias Soeken. Improved quantum circuits for elliptic curve discrete logarithms. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020, Paris, France, April 15-17, 2020, Proceedings*, volume 12100 of *Lecture Notes in Computer Science*, pages 425–444. Springer, 2020.

[HPS+19] Sabrina S. Hong, Alexander T. Papageorge, Prasahnt Sivarajah, Genya Crossman, Nicolas Didier, Anthony M. Polloreno, Eyob A. Sete, Stefan W. Turkowski, Marcus P. da Silva, Blake R. Johnson. Demonstration of a parametrically-activated entangling gate protected from flux noise. arXiv:1901.08035, preprint quantum-ph/0601097, 2004.

[HRS17] Thomas Haener, Martin Roetteler, and Krysta M. Svore. Factoring using $2n + 2$ qubits with toffoli based modular multiplication. *Quantum Information and Computation*, 18(7-8):673–684, 2017.

[HSA+16] T. P. Harty, M. A. Sepiol, D. T. C. Allcock, C. J. Ballance, J. E. Tarlton, and D. M. Lucas. High-fidelity trapped-ion quantum logic using near-field microwaves. *Phys. Rev. Lett.*, 117:140501, 2016.

[IOK+12] 石坂智，小川朋宏，河内亮周，木村元，林正人．量子情報科学入門．共立出版，2012.

[JBM+18] Shuxian Jiang, Keith A. Britt, Alexander J. McCaskey, Travis S. Humble, and Sabre Kais. Quantum annealing for prime factorization. *Scientific Reports*, 8(17667), 2018.

[KGA+11] A. Keselman, Y. Glickman, N. Akerman, S. Kotler, and R. Ozeri. High-fidelity state detection and tomography of a single-ion zeeman qubit. *New Journal of Physics*, 13(7):073027, 2011.

[KSB+20] Morten Kjaergaard, Mollie E. Schwartz, Jochen Braumüller, Philip Krantz, Joel I.-J. Wang, Simon Gustavsson, and William D. Oliver. Superconducting qubits: Current state of play. *Annual Review of Condensed Matter Physics*, 11(1):369–395, 2020.

[KSG+20] Morten Kjaergaard, Mollie E. Schwartz, Ami Greene, Gabriel O. Samach, Andreas Bengtsson, Michael O'Keeffe, Christopher M. McNally, Jochen Braumüller, David K. Kim, Philip Krantz, Milad Marvian, Alexander Melville, Bethany M. Niedzielski, Youngkyu Sung, Roni Winik, Jonilyn Yoder, Danna Rosenberg, Kevin Obenland, Seth Lloyd, Terry P. Orlando, Iman Marvian, Simon Gustavsson, William D. Oliver. Programming a quantum computer with quantum instructions. *arXiv: Quantum Physics*, arXiv:2001.08838, 2020.

[Kun05] Noboru Kunihiro. Exact analyses of computational time for factoring in quantum computers. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 88-A(1):105–111, 2005.

[LBC+12] E. Lucero, R. Barends, Y. Chen, J. Kelly, M. Mariantoni, A. Megrant, P. O'Malley, D. Sank, A. Vainsencher, J. Wenner, T. White, Y. Yin, A. N. Cleland, and J. M. Martinis. Computing prime factors with a Josephson phase qubit quantum processor. *Nature Physics*, 8:719–723, 2012.

[LBYP07] Chao-Yang Lu, Daniel E. Browne, Tao Yang, and Jian-Wei Pan. Demonstration of a compiled version of Shor's quantum factoring algorithm using photonic qubits. *Phys. Rev. Lett.*, 99, 2007.

[Lit19] Daniel Litinski. A game of surface codes: Large-scale quantum computing with lattice surgery. *Quantum*, 3:128, 2019.

[LJMP90] Arjen K. Lenstra, Hendrik W. Lenstra Jr., Mark S. Manasse, and John M. Pollard. The number field sieve. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 564–572, 1990.

[LWL+07] B. P. Lanyon, T. J. Weinhold, N. K. Langford, M. Barbieri, D. F. V. James, A. Cilchrist, and A. G. White. Experimental demonstration of a compiled version of Shor's algorithm with quantum entanglement. *Phys. Rev. Lett.*, 99, 2007.

[MFM+16] David C. McKay, Stefan Filipp, Antonio Mezzacapo, Easwar Magesan, Jerry M. Chow, and Jay M. Gambetta. Universal gate for fixed-frequency qubits via a tunable bus. *Phys. Rev. Applied*, 6:064007, 2016.

[MLLL+12] E. Martin-Lopez, A. Laing, T. Lawson, R. Alvarez, X.-Q. Zhou, and J. L. O'Brien. Experimental realisation of Shor's quantum factoring algorithm using qubit recycling. *Nature Photon*, 6:773–776, 2012.

[MNM+16] T. Monz, D. Nigg, E. A. Martinez, M. F. Brandl, P. Schindler, R. Rines, S. X. Wang, I. L. Chuang, and R. Blatt. Realization of a scalable Shor algorithm. *Science*, 351:1068–1070, 2016.

[NC11] Michael A. Nielsenand and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, USA, 10th edition, 2011.

[NIST13] NIST. FIPS: 186-4: Digital Signature Standard (DSS). 2013.

[OC17] Joe O'Gorman and Earl T. Campbell. Quantum computation with realistic magic-state factories. *Phys. Rev. A*, 95:032338, 2017.

[OWC+11] C. Ospelkaus, U. Warring, Y. Colombe, K. R. Brown, J. M. Amini, D. Leibfried, and D. J. Wineland. Microwave quantum logic gates for trapped ions. *Nature*, 476:181–184, 2011.

[PGM+12] S. Poletto, Jay M. Gambetta, Seth T. Merkel, John A. Smolin, Jerry M. Chow, A. D. Córcoles, George A. Keefe, Mary B. Rothwell, J. R. Rozen, D. W. Abraham, Chad Rigetti, and M. Steffen. Entanglement of two superconducting qubits in a waveguide cavity via monochromatic two-photon excitation. *Phys. Rev. Lett.*, 109:240505, 2012.

[PMO09] A. Politi, J. C. F. Matthews, and J. L. O'Brien. Shor's quantum factoring algorithm on a photonic chip. *Science*, 325:1221, 2009.

[PMS+16] Hanhee Paik, A. Mezzacapo, Martin Sandberg, D. T. McClure, B. Abdo, A. D. Córcoles, O. Dial, D. F. Bogorin, B. L. T. Plourde, M. Steffen, A. W. Cross, J. M. Gambetta, and Jerry M. Chow. Experimental demonstration of a resonator-induced phase gate in a multiqubit circuit-qed system. *Phys. Rev. Lett.*, 117:250502, 2016.

[RGR+18] S. Rosenblum, Y. Y. Gao, P. Reinhold, C. Wang, C. J. Axline, L. Frunzio, S. M. Girvin, Liang Jiang, M. Mirrahimi, M. H. Devoret, and R. J. Schoelkopf. A CNOT gate between multiphoton qubits encoded in two cavities. *Nature Communications*, 9, 652, 2018.

[RNSL17] Martin Roetteler, Michael Naehrig, Krysta M. Svore, and Kristin E. Lauter. Quantum resource estimates for computing elliptic curve discrete logarithms. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Proceedings, Part II*, volume 10625 of *Lecture Notes in Computer Science*, pages 241–270. Springer, 2017.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[SBT+18] V. M. Schäfer, C. J. Ballance, K. Thirumalai, L. J. Stephenson, T. G. Ballance, A. M. Steane, D. M. Lucas. Fast quantum logic gates with trapped-ion qubits. *Nature*, 555(7694):75-78, 2018.

[Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science*, pages 124–134. IEEE Computer Society, 1994.

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.

[SMCG16] Sarah Sheldon, Easwar Magesan, Jerry M. Chow, and Jay M. Gambetta. Procedure for systematically tuning up cross-talk in the cross-resonance gate. *Phys. Rev. A*, 93:060302, 2016.

[SSV13] John A. Smolin, Graeme Smith, and Alexander Vargo. Oversimplifying quantum factoring. *Nature*, 499:163–165, 2013.

[TGL+15] T. R. Tan, J. P. Gaebler, Y. Lin, Y. Wan, R. Bowler, D. Leibfried, and D. J. Wineland. Multi-element logic gates for trapped-ion qubits. *Nature*, 528:380–383, 2015.

[VBE96] Vlatko Vedral, Adriano Barenco, and Artur Ekert. Quantum networks for elementary arithmetic operations. *Phys. Rev. A*, 54:147–153, 1996.

[VSB+01] L. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood, and I. L. Chuang. Experimental realization of shor's quantum factoring algorithm using nuclear magnetic resonance. *Nature*, 414:883–887, 2001.

[WRW+16] S. Weidt, J. Randall, S. C. Webster, K. Lake, A. E. Webb, I. Cohen, T. Navickas, B. Lekitsch, A. Retzker, and W. K. Hensinger. Trapped-ion quantum logic with global radiation fields. *Phys. Rev. Lett.*, 117:220501, 2016.

[Zal98] Christof Zalka. Fast versions of Shor's quantum factoring algorithm. arXiv preprint quantum-ph/9806084, 1998.

[Zal06] Christof Zalka. Shor's algorithm with fewer (pure) qubits. arXiv preprint quantum-ph/0601097, 2006.