Universidade de Aveiro Departamento de Física
2018

Jorge Manuel Tavares Brandão

Ataques quânticos e os criptossistemas de McEliece

Jorge Manuel
Tavares Brandão

Ataques quânticos e os criptossistemas de McEliece

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Física, realizada sob a orientação científica da Professora Doutora *Margarida Maria Resende Vieira Facão*, Professora Auxiliar do Departamento de Física da Universidade de Aveiro, do Professor Doutor *Diego Oscar Napp Avelli*, Investigador Auxiliar do Departamento de Matemática da Universidade de Aveiro, e do Professor Doutor *Paulo José Fernandes Almeida*, Professor Auxiliar do Departamento de Matemática da Universidade de Aveiro.

o júri / the jury

presidente / president

Prof. Doutor Leonel Marques Vitorino Joaquim
Professor Auxiliar da Universidade de Aveiro

vogais / examiners committee

Profª. Doutora Margarida Maria Resende Vieira Facão
Professora Auxiliar da Universidade de Aveiro (orientador)

Prof. Doutor Diego Oscar Napp Avelli
Investigador Auxiliar da Universidade de Aveiro (co-orientador)

Prof. Doutor Paulo José Fernandes Almeida
Professor Auxiliar da Universidade de Aveiro (co-orientador)

Prof. Doutor José Pedro Miranda Mourão Patrício
Professor Auxiliar da Universidade Minho (arguente)

agradecimentos / acknowledgements

Em primeiro lugar agradeço à minha família, principalmente aos meus pais e avós, pelo constante apoio, força e amor que me deram, desde sempre.
Agradeço aos meus amigos que estiveram sempre presentes, alguns mesmo nem estando fisicamente cá. Sempre me acompanharam, apoiaram e ajudaram.
Por fim, mas não menos importantes nesta etapa, agradeço à professora Margarida, ao professor Paulo e ao professor Diego, por toda a ajuda que me deram, por toda a disponibilidade que sempre tiveram, por desde o início me terem feito sentir entre amigos e me terem dado o à vontade para questionar e discutir todos e qualquer assunto. Pela paciência sobrehumana que, perante algumas (muitas) dessas discussões, certamente precisaram ter. Pela boa disposição que demonstraram a cada momento. Obrigado por me terem proporcionado o maior momento de aprendizagem deste meu percurso académico.
A todos, um muito obrigado.

Palavras-chave RSA, Entrelaçamento quântico, Paralelismo quântico, QFT, Decoerência, Algoritmo de Shor, ISD, Criptossistema de McEliece.

Resumo O algoritmo de Shor e a evolução da computação quântica trouxeram grandes ameaças à segurança dos criptossistemas atuais. Pela necessidade de se encontrar novas alternativas, surgiu a criptografia pós-quântica.
Esta dissertação pode ser dividida em três partes principais. Na primeira apresentamos alguns elementos de teoria dos números e descrevemos um criptossistema clássico, o RSA, cuja segurança pode ser facilmente quebrada recorrendo a ataques quânticos. Na segunda parte descrevemos a implementação do algoritmo de Shor, que é um ataque a esse e a todos os criptossistemas cuja segurança reside no problema da fatorização ou no problema do logaritmo discreto. Na implementação deste algoritmo minimizámos o número de qubits necessários. Para isso, utilizámos transformadas de Fourier quânticas para realizar operações como a adição, a multiplicação e a exponenciação modular. Assim, segundo a nossa abordagem, precisámos de apenas $L_1 + 2L_2 + 3$ qubits, para implementar o algoritmo de Shor, ou seja, para encontrar o período da função $f(x) = a^x \bmod N$, onde L_1 corresponde ao número de qubits necessários para representar x tal que $N^2 < 2^{L_1} < 2N^2$ e L_2 corresponde ao número de qubits necessários para representar N. Por fim, na última parte desta dissertação, apresentaremos elementos da criptografia pós-quântica, em particular, um protocolo proposto por nós. Este é uma nova variante do criptossistema de McEliece, no qual se propõe o uso de códigos convolucionais em vez do uso de códigos de bloco. Assim, a parte codificadora da nossa chave pública, que denotaremos por $G'(D)$, será ela própria convolucional. Ao estudarmos várias possibilidades de ataques para este criptossistema, verificámos que, em comparação com as restantes variantes, para uma chave pública menor conseguimos ter um fator de trabalho muito maior, o que se traduz numa maior segurança.

Keywords RSA, Quantum entanglement, Quantum paralelism, QFT, Decoherence, Shor's algorithm, ISD, McEliece cryptosystem.

Abstract Shor's algorithm and the evolution of quantum computing brought big threats to the security of the current cryptosystems. The need to find new alternatives created what is known as post-quantum cryptography.
This dissertation can be splitted into three main parts. In the first one, we present elements of number theory and we describe a classical cryptosystem, the RSA, whose security can be easily broken using quantum attacks. In the second part, we describe the implementation of Shor's algorithm, which is an attack to RSA but also to all the cryptosystems whose safety lies in the factorization problem or in the discrete logarithm problem. In the implementation of this algorithm we minimized the number of qubits required. For this, we used quantum Fourier transforms to perform operations such as addition, multiplication and modular exponentiation. Thus, according to our approach, we need only $L_1 + 2L_2 + 3$ qubits to implement the Shor's algorithm, that is, to find the period of the function $f(x) = a^x \bmod N$, where L_1 is the number of bits needed to represent x such that $N^2 < 2^{L_1} < 2N^2$, and L_2 is the number of bits needed to represent N. Finally, in the last part of this dissertation, we present elements of post-quantum cryptography, in particular, a new protocol proposed by us. This is a new variant of the McEliece cryptosystem, on which we propose the use of convolutional codes instead of block codes. Thus, the encoder part of our public key, which we will denote by $G'(D)$, will be itself convolutional. By studying several possibilities of attacks for this cryptosystem, we verified that, compared to the other variants, for a smaller public key we can have a much larger work factor, which can be translated in a greater security.

Lista de conteúdos

Lista de conteúdos **i**

Lista de figuras **iii**

Lista de tabelas **v**

Lista de acrónimos **vii**

1 Introdução **1**

2 Preliminares **5**

2.1 Teoria dos Números 5

2.1.1 Algoritmo de Euclides 5

2.1.2 Função φ de Euler 6

2.1.3 Teorema Chinês dos Restos 6

2.1.4 Lagrange, Euler e Fermat 7

2.1.5 Exponenciação modular rápida 7

2.2 Criptografia pré-quântica 7

2.2.1 P versus NP 8

2.2.2 RSA 8

3 Computação Quântica **11**

3.1 Fundamentos da Física Quântica 11

3.1.1 Produto interno e espaço de Hilbert 11

3.1.2 Postulados 12

3.1.3 Princípio da Não Clonagem 13

3.2 Fundamentos da Computação Quântica 13

3.2.1 Bits e Qubits 13

3.2.2 Esfera de Bloch 14

3.2.3 Portas lógicas quânticas 14

3.2.4 Portas lógicas quânticas de um único qubit 15

3.2.5 Entrelaçamento quântico e paralelismo quântico 18

3.3 Implementações práticas e decoerência 19

3.4 Circuitos quânticos 19

3.4.1 Transformada de Fourier Quântica 20

3.4.2 Adição e multiplicação quântica 21

3.5 Algoritmo de Shor 25

3.5.1 Algoritmo - método clássico . . . 25
3.5.2 Algoritmo - parte quântica . . . 26
3.6 Conclusões . . . 31

4 Criptografia Pós-Quântica **33**
4.1 Criptografia baseada em Códigos . . . 33
4.1.1 Teoria dos Códigos . . . 33
4.1.2 Descodificação por síndrome . . . 37
4.1.3 Códigos Reed-Solomon . . . 37
4.1.4 Códigos Convolucionais . . . 38
4.2 Criptossistema de McEliece . . . 39
4.2.1 Descrição . . . 39
4.3 Limitações e Ataques . . . 39
4.3.1 Descodificação por Conjunto de Informação . . . 40
4.4 Nova variante do McEliece PKC baseada em códigos convolucionais . . . 41
4.4.1 Descrição . . . 41
4.4.2 Segurança e exemplos numéricos . . . 43
4.4.3 Exemplo prático . . . 45
4.5 Conclusões . . . 47

Referências **49**

Lista de figuras

1.1 Sistema de troca de mensagens cifradas entre Alice e Bob, quando intercetadas por um agente não autorizado (Eve). 2
1.2 Computador quântico do IBM em a) [6] e respetivos componentes em b) [7]. Processador quântico de 72 qubits Bristlecone, da Google, em c) [8]. 3

3.1 Representação do estado de um qubit $|\psi\rangle$ como um ponto na superfície da Esfera de Bloch. 15
3.2 Forma geral de uma porta quântica para um único qubit. 15
3.3 Símbolo correspondente à medição. 15
3.4 Portas lógicas de Pauli e respetiva representação simbólica e matriz. 16
3.5 Porta quântica Hadamard e respetiva representação matricial. 16
3.6 Porta quântica de fase (duas notações equivalentes). 17
3.7 Porta quântica CNOT e respetiva representação matricial. 17
3.8 Porta quântica SWAP, circuito equivalente com portas CNOT e respetiva representação matricial. 17
3.9 (a) Porta quântica arbitrária U-controlada. (b) Circuito equivalente para uma fase R_l-controlada. 18
3.10 Porta quântica Toffoli e respetiva representação matricial. 18
3.11 Exemplo de um circuito onde ocorre entrelaçamento quântico. 19
3.12 Esquema da disposição dos qubits no computador quântico, *IBM Q 5 Tenerife*, do IBM. 20
3.13 Representação geral do circuito da Transformada de Fourier Quântica. 21
3.14 Adição quântica usando QFTs. 23
3.15 Representação da porta lógica quântica da subtração assim como os possíveis estados de saída. 23
3.16 Representação do circuito e da porta lógica quântica para a adição de $a \bmod N$. a) soma a e subtrai N; b) verifica se $a+b > N$; c) soma N se $a+b < N$; d) coloca o qubit auxiliar a $|0\rangle$; e) reverte resultado após c) no registo $|b\rangle$. 24
3.17 Circuito quântico para encontrar a ordem $\mathrm{ord}_N(a)$, $f(x)$ implementa $a^x \bmod N$. . . . 27
3.18 Circuito quântico do algoritmo de Shor, simplificado para $N = 15$ e $a = 4$. 32

4.1 Exemplo de dois códigos: o de a) tem distância mínima inferior a $2t+1$ e o de b) tem distância mínima maior ou igual que $2t+1$. 37
4.2 Alguns algoritmos da classe de ataques ISD e respetiva distribuição de erros. 40

Lista de tabelas

4.1 Soma e multiplicação entre elementos de um corpo finito $GF(2^2)$. 34
4.2 Formas de representação de um corpo $GF(8)$. 35
4.3 Comparação entre o McEliece PKC de Niederreiter e a nossa proposta. 45

Lista de acrónimos

DFT	*Discrete Fourier Transform*
DSA	*Digital Signature Algorithm*
ECDSA	*Elliptic Curve Digital Signature Algorithm*
ETSI	*European Telecommunications Standards Institute*
ISD	*Information Set Decoding*
mdc	*máximo divisor comum*
PKC	*Public Key Cryptosystem*
QFT	*Quantum Fourier Transform*
RSA	*Rivest-Shamir-Adleman*
WF	*Work Factor*

Capítulo 1

Introdução

All understanding begins with our not accepting the world as it appears.

Alan Kay

Os sistemas de comunicação podem ser clássicos ou quânticos. Os clássicos são aqueles que podem ser descritos usando uma formulação clássica, semi-clássica ou quântica enquanto os sistemas quânticos são aqueles que só podem ser descritos usando uma formulação quântica. Estes sistemas, regidos por leis quânticas, permitem a realização de tarefas que não são realizadas ou são ineficientemente realizadas nos sistemas baseados nas leis clássicas. No entanto, apesar destes sistemas alternativos aos clássicos serem cada vez mais um alvo de estudo, porque com eles consegue-se, por exemplo, saber se uma comunicação foi ou não intercetada, como no caso do protocolo BB84 [1] usado em criptografia para distribuição quântica de chaves, estes não são ainda muito práticos. Ainda relativamente ao BB84, ele necessita de equipamento muito dispendioso e só é funcional para curtas distâncias. Assim, ao longo desta dissertação, iremos considerar que todos os sistemas de comunicação são clássicos.

A evolução tecnológica coloca-nos num mundo cada vez mais digital tornando, assim, de extrema importância a segurança nas comunicações assim como a transmissão segura de dados. O ramo que se encarrega de garantir esta segurança chama-se *criptografia*. Este, juntamente com a *criptoanálise*, são os dois ramos da *criptologia*. Deste modo, podemos definir *criptografia* como a arte de trocar mensagens (comunicar) tornando-as indecifráveis para qualquer entidade não autorizada e, *criptoanálise*, como o estudo da segurança dos criptossistemas. Desta advém o estudo dos possíveis ataques e a tentativa de precaver que estes aconteçam, melhorando, deste modo, a segurança do sistema. Vamos supor que a Alice quer usar um sistema de comunicação para enviar uma mensagem digital, de forma secreta, para o Bob. No entanto, ela receia que esta possa ser intercetada por algum espião (que chamaremos de Eve). Que poderá a Alice fazer para enviar a sua mensagem de forma segura? Supondo que Eve interceta a mensagem, ou seja, recebe a mesma mensagem que Bob, como poderá a Alice garantir que apenas o Bob conseguirá aceder à informação da mensagem? A primeira criptografia a surgir foi a de chave privada, também denominada por *simétrica*. Observemos a figura 1.1. Nesta a Alice cifra uma mensagem usando uma chave secreta e envia para o Bob. O Bob, recebe a mensagem e usa uma chave secreta para a decifrar. A *criptografia simétrica*, é aquela em que a chave que a Alice usa para cifrar a mensagem é a mesma que o Bob usa para a decifrar. A principal limitação deste tipo de criptografia, além de o Bob necessitar de ter uma chave secreta diferente para cada pessoa que lhe quisesse enviar uma mensagem cifrada, residia na distribuição de chaves, pois estas, nos sistemas de comunicação clássicos, podem ser facilmente intercetadas sem que ninguém perceba. No entanto, na década de 70, foram criados sistemas de criptografia clássicos de chave pública. Estes sistemas usam uma chave pública, a que todos têm acesso e que utilizam para cifrar a informação, e uma chave privada para a decifrar. Este tipo de criptografia, de chave pública, foi chamada de *assimétrica*. A segurança deste tipo de sistema reside no facto de a chave pública, que é composta pelos elementos da chave privada, mesmo computacionalmente, ser muito difícil de decompor, ou seja, os algoritmos clássicos que existem para obter a chave privada a partir da chave pública não são polinomiais. Como bom exemplo deste tipo de sistema temos o criptossistema *Rivest-Shamir-Adleman* (RSA), no qual a chave pública não é mais que a multiplicação de dois

números primos muito grandes. Este criptossistema irá ser explicado de forma detalhada mais à frente nesta dissertação. Neste, apenas aqueles que conhecem a chave privada, ou seja, sabem quais os primos que fatorizam o número da chave pública, podem decifrar a mensagem. Quem tem acesso apenas à chave pública, só consegue cifrar. Este tipo de criptossistema veio resolver o problema da distribuição de chaves pois, nestes casos, não há chaves a distribuir secretamente. No entanto, com a evolução tecnológica e o surgimento da computação quântica, a segurança de alguns sistemas clássicos assimétricos, como é caso do RSA, ElGamal, Algoritmo de Assinatura digital (*Digital Signature Algorithm* (DSA)) ou Algoritmo de Assinatura Digital de Curvas Elípticas (*Elliptic Curve Digital Signature Algorithm* (ECDSA)), voltou a ser ameaçada. O problema da fatorização e o problema do algoritmo discreto, que eram considerados problemas difíceis, ou seja, não possuíam um algoritmo clássico que os resolvesse em tempo polinomial foram, na década de 90, resolvidos por um algoritmo de fatorização que é polinomial quando implementado num computador quântico. A este algoritmo, proposto por Peter Shor [2], deu-se o nome de *algoritmo de Shor*.

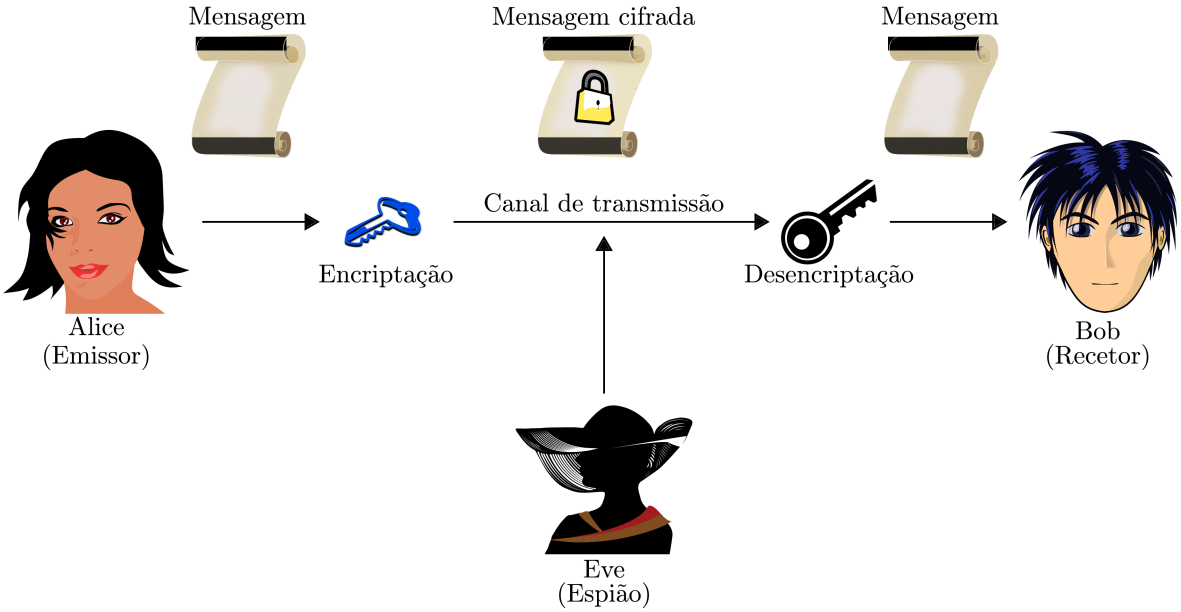


Figura 1.1: Sistema de troca de mensagens cifradas entre Alice e Bob, quando intercetadas por um agente não autorizado (Eve).

Apesar da descoberta do algoritmo de Shor, os computadores quânticos ainda não estão no nível necessário para que este seja facilmente implementado. Existem várias limitações que se traduzem, entre outras coisas, numa perda rápida da coerência dos qubits e em várias dificuldades no que toca à implementação de algoritmos. Apenas em 2012 um computador quântico conseguiu fatorizar o número 15 usando o algoritmo de Shor (ver [3]). No entanto, em junho de 2015, o Instituto Europeu de Normas de Telecomunicações, *European Telecommunications Standards Institute* (ETSI), alertou as organizações que precisam de guardar informações ou protegerem a privacidade de transações online, para mudarem a sua segurança para técnicas de encriptação que sejam quanticamente seguras, destacando o facto de que no futuro os computadores quânticos iriam ser capazes de fatorizar grandes inteiros e assim quebrar a maioria das cifras que neste momento são utilizadas em repositórios online, *websites*. Neste alerta, o instituto também referiu que até a informação cifrada que já permanece guardada há mais de 25 anos, seria um alvo fácil para aqueles que possuírem acesso a este tipo de computadores (ver [4]).

Este aviso do ETSI não foi o primeiro sinal de que a era dos computadores quânticos poderia estar próxima. Já há alguns anos, que se tem observado grandes investimentos por parte de gigantes tecnológicas como a Intel, Google, IBM e mais recentemente a Microsoft, na área da computação quântica. O Reino Unido e a União Europeia, entretanto, também anunciaram dois programas prioritários de investimento, um no valor de mais de 300 milhões de euros e outro no valor de mais

de 1100 milhões de euros, para o desenvolvimento e comercialização de tecnologias quânticas (ver [5]). Após tudo isto, o professor Sir Peter Knight, do Colégio Imperial de Londres, comentou que todo este interesse por parte da indústria tecnológica, na computação quântica, é mais que bem fundamentado, pois, os circuitos quânticos estão quase num nível em que a sua evolução pode escalar muito rapidamente, podendo assim ser utilizados na construção de equipamentos quânticos poderosos. Ele estima que o primeiro computador quântico funcional, ou seja, poderoso e com estabilidade suficiente para que se consiga facilmente implementar algoritmos complexos, poderá aparecer durante a próxima década. A evolução dos computadores quânticos nos últimos anos tem sido enorme. Ainda em 2016 os computadores quânticos mais desenvolvidos possuíam apenas 12 qubits enquanto que no ano seguinte, a IBM anuncia um computador de 50 qubits e, ainda este ano, a Intel já apresentou um chip quântico de 49 qubits, enquanto que a Google superou a líder IBM, estabelecendo até à data o recorde de computador com mais qubits, com o seu novo processador quântico de 72 qubits, a que deu o nome de *Bristlecone*. Na figura 1.2 podemos observar em a), um computador quântico da IBM, em b), o seu processador e outras componentes, e em c), o Bristlecone da Google.

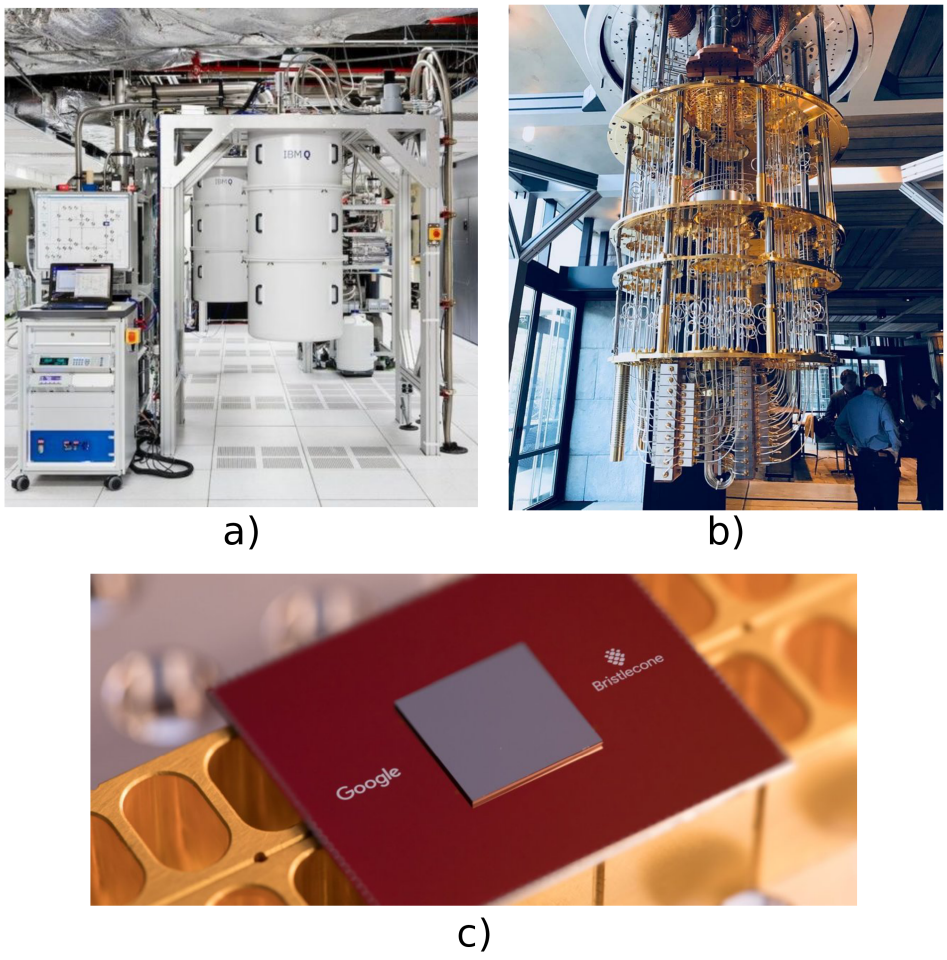


Figura 1.2: Computador quântico do IBM em a) [6] e respetivos componentes em b) [7]. Processador quântico de 72 qubits Bristlecone, da Google, em c) [8].

Deste modo, antes que a computação quântica atinja um nível suficiente alto para conseguir quebrar com facilidade os sistemas de segurança atuais, existe a necessidade de estudar e desenvolver sistemas alternativos resistentes a ataques quânticos, nomeadamente ao algoritmo de Shor. Com a finalidade de encontrar e estudar essas alternativas, surge a Criptografia Pós-quântica. Antes de introduzirmos os criptossistemas pós-quânticos, convém deixar uma nota sobre a designada *criptografia quântica*. A criptografia quântica, da qual o já referido protocolo BB84 é um exemplo, pode pensar-se como uma alternativa aos sistemas clássicos. Esta é constituída por protocolos de troca de chaves capazes de

nos dar informação sobre se a troca foi ou não intercetada por algum agente não autorizado. Estas chaves seriam posteriormente utilizadas em criptografia simétrica. No entanto, pelos inconvenientes inicialmente referidos acerca destes sistemas, não iremos aqui considerar este tipo de criptografia como a melhor alternativa aos sistemas clássicos atuais.

Entre os criptossistemas pós-quânticos mais promissores para substituírem os criptossistemas de chave pública (*Public Key Cryptosystem* (PKC)) atuais, estão os baseados em códigos corretores de erros, nomeadamente o criptossistema de McEliece [9]. Como será aprofundado mais à frente nesta dissertação, quando uma mensagem é enviada por um canal ruidoso, podem ocorrer erros. Como tal, é essencial ter um mecanismo capaz de detetar e corrigir esses erros. Daí surgiram os primeiros códigos corretores de erros, já na década de 40, quando Richard Hamming desenvolveu um código capaz de detetar até dois erros e corrigir um, se este fosse único. Assim, as mensagens passaram a ser codificadas com estes códigos antes de serem enviadas, e descodificadas quando recebidas. Mais tarde, em 1978, Robert McEliece, apresentou o primeiro criptossistema baseado na teoria dos códigos. Este utilizava códigos Goppa binários, pois, estes possuíam um algoritmo rápido de descodificação. A sua segurança era baseada no facto de ser muito difícil descodificar um código linear aleatório.

O criptossistema de McEliece tem recebido muita atenção ao longo dos últimos anos por, até ao momento, ser capaz de resistir aos ataques quânticos. No entanto, este possui algumas desvantagens, como baixa taxa de encriptação e tamanho da chave muito grande, o que em parte se deve ao facto deste ser baseado em códigos Goppa. Assim, com a intenção de mitigar estas desvantagens, muitas outras variantes deste criptossistema foram surgindo. Umas usavam outro tipo de códigos lineares de bloco, mais pequenos, como por exemplo os Reed-Solomon, e uma outra, descrita em [10], sugeria o uso de códigos convolucionais, mas foi rapidamente quebrada. Nos códigos de bloco, a informação é dividida em blocos de tamanho fixo e a codificação é feita bloco a bloco, usando para cada bloco o mesmo procedimento, sem que seja necessária qualquer informação sobre o bloco anterior. Nos códigos convolucionais, a informação é na mesma dividida em blocos mas, ao mesmo tempo é vista como toda uma sequência, ou seja, a informação é na mesma codificada bloco a bloco, mas, a codificação de cada bloco depende da informação contida num ou mais blocos imediatamente anteriores. Assim, podemos dizer que a grande diferença entre os códigos de bloco e os convolucionais é que estes últimos têm memória.

Motivados pelo facto de a computação quântica ser cada vez mais uma ameaça aos sistemas de segurança atuais iremos, nesta dissertação, apresentar e fazer uma criptoanálise de uma alternativa a estes. Esta é uma nova variante do criptossistema de McEliece, desenvolvida por nós, baseada em códigos convolucionais. No entanto, antes disso, iremos dar a conhecer um pouco a criptografia clássica, onde descreveremos o criptossistema de RSA, e de seguida iremos descrever e implementar o algoritmo de Shor quântico, que é o melhor ataque quântico, que se conhece, aos criptossistemas atuais cuja segurança reside num problema de fatorização.

Como cada capítulo desta dissertação aborda uma área diferente, as conclusões serão apresentadas a cada capítulo em vez de numa conclusão final geral. Parte dos resultados que iremos aqui apresentar, nomeadamente acerca da nova variante do McEliece PKC, já foram apresentados através de um poster com o tema *McEliece Cryptosystem Based on Convolutional Codes* na conferência *5thICCDS'2018*[1], e com um artigo no congresso internacional de *Álgebra Lineal, Análisis Matricial y Aplicaciones* em Alama (ver [11]). Para a parte dos ataques quânticos, usámos a linguagem de programação python 3.5.4 e a biblioteca *qiskit* disponibilizada pela IBM, para criar um programa que simula a adição *drapper*, a adição modular quântica, transformadas de Fourier quânticas e respetivas inversas, e por último, o algoritmo de Shor quântico para a fatorização do número 15. Para a parte da criptografia pós-quântica, escrevemos também vários programas, desde a implementação da nossa variante do criptossistema de McEliece (para um código de Hamming e para códigos Reed-Solomon), até ao cálculo do fator de trabalho destes criptossistemas, para ataques da classe Descodificação por Conjunto de Informação, *Information Set Decoding* (ISD). Os códigos são de acesso livre e podem ser consultados, os que dizem respeito ao capítulo dos ataques quânticos, em [12], e os que dizem respeito ao capítulo da criptografia pós-quântica, em [13].

[1] The 5th International Conference on Complex Dynamical Systems in Life Sciences: Modeling and Analysis, 2018

Capítulo 2

Preliminares

-How long do you want these messages to remain secret?[...]-I want them to remain secret for as long as men are capable of evil.

Neal Stephenson

A *Criptografia* consiste no desenvolvimento e uso de métodos que permitem que uma comunicação seja realizada de forma segura ou privada (secreta) mesmo quando intercetada por terceiros, ou seja, consiste no desenvolvimento de técnicas que permitem que se um intruso intercetar uma mensagem cifrada este seja incapaz de descobrir a mensagem original. *Criptografia clássica* é aquela em que esses métodos e protocolos tem como base princípios não quânticos.

Neste capítulo iremos descrever um criptossistema clássico, o RSA. Como tal, começaremos por introduzir algumas noções de teoria dos números, necessárias à sua compreensão, assim como para a compreensão de alguns tópicos posteriormente abordados nesta dissertação.

2.1 Teoria dos Números

A teoria dos números é o ramo da matemática que estuda as propriedades dos números inteiros, dando principal enfoque aos números primos e à sua fatorização. Ao longo desta secção iremos apresentar algumas noções sobre esta vasta área da matemática.

2.1.1 Algoritmo de Euclides

O *algoritmo de Euclides* é um processo simples e eficiente de encontrar o *máximo divisor comum* (mdc) entre quaisquer dois inteiros, a e b, diferentes de zero. Este elemento pode ser denotado por $\text{mdc}(a, b)$ ou simplesmente por (a, b).

Consideremos a e b inteiros positivos. Então existem dois inteiros, q_0 e r_0, tais que

$$a = q_0 b + r_0 \tag{2.1}$$

com $0 \leq r_0 < b$. Se $r_0 \neq 0$ então, do mesmo modo, existem dois inteiros, q_1 e r_1 , tais que

$$b = q_1 r_0 + r_1 \tag{2.2}$$

com $0 \leq r_1 < b$. Podemos continuar com este processo até obtermos um resto da divisão $r_{k+1} = 0$.

Assim, uma esquematização do algoritmo de Euclides seria

$$\begin{cases} a = q_0 b + r_0 \\ b = q_1 r_0 + r_1 \\ r_0 = q_2 r_1 + r_2 \\ \vdots \\ r_{k-2} = q_k r_{k-1} + r_k \\ r_{k-1} = q_{k+1} r_k \end{cases}$$

onde r_k corresponde ao $\text{mdc}(a, b)$.

O algoritmo de Euclides também permite encontrar inteiros u e v tais que

$$au + bv = (a, b). \tag{2.3}$$

Este resultado pode ser utilizado para provar o lema seguinte.

Lema: Se $1 < a < n$ e $\text{mdc}(a, n) = 1$ então existe um único b tal que $1 < b < n$ e $ab = 1 \bmod n$.

2.1.2 Função φ de Euler

Seja $n \geq 1$. O número de inteiros positivos, menores ou iguais a n e coprimos com n, é denotado por $\varphi(n)$. A função $\varphi(n)$ é denominada de função φ de Euler. Esta é multiplicativa, isto é, se $(a, b) = 1$ então $\varphi(ab) = \varphi(b)\varphi(b)$ (ver [14]).

Vamos supor que $n > 1$. Então n pode ser fatorizado em primos ficando

$$n = p_1^{a_1} p_2^{a_2} \ldots p_k^{a_k}, \tag{2.4}$$

com $a_1, a_2, \ldots, a_k \geq 1$. Então a função de Euler para n é

$$\varphi(n) = \varphi(p_1^{a_1})\varphi(p_2^{a_2}) \ldots \varphi(p_k^{a_k}), \tag{2.5}$$

onde (ver [15])

$$\varphi(p^a) = p^a - p^{a-1} = p^a \left(1 - \frac{1}{p}\right). \tag{2.6}$$

2.1.3 Teorema Chinês dos Restos

Em teoria dos números, dizemos que um inteiro x é congruente com o inteiro a módulo m quando m divide $x - a$ $(m|(x - a))$. Neste caso escrevemos $x \equiv a \bmod m$. O *Teorema Chinês dos Restos* é um método rápido e eficiente para resolver certos sistemas de congruências. Vamos supor que temos $m_1, m_2, \ldots, m_k$ inteiros positivos e coprimos entre si 2 a 2. Vamos supor também que $a_1, a_2, \ldots, a_k$ são inteiros. Então, este teorema afirma que o sistema

$$\begin{cases} x \equiv a_1 \bmod m_1 \\ x \equiv a_2 \bmod m_2 \\ \vdots \\ x \equiv a_k \bmod m_k \end{cases}$$

tem uma única solução módulo $M = m_1 m_2 \ldots m_k$. O seu algoritmo é o seguinte: considere-se $M_i = \frac{M}{m_i}$ e $y_i = M_i^{-1} \bmod m_i$ para $1 \leq i \leq k$. Então a única solução do sistema é

$$x = \sum_{i=1}^{k} a_i M_i y_i \bmod M. \tag{2.7}$$

2.1.4 Lagrange, Euler e Fermat

Sejam a e n inteiros tais que $\text{mdc}(a,n) = 1$. Então define-se como *ordem* de $a \bmod n$, o menor inteiro positivo b tal que

$$a^b \equiv 1 \bmod n \tag{2.8}$$

e denotamo-la por $\text{ord}_n(a)$ (ver [15]).

Teorema de Lagrange: Seja G um grupo multiplicativo com n elementos e seja $g \in G$. Então $\text{ord}_n(g) \mid n$.

Pequeno teorema de Fermat: Seja p um primo e $a \in \mathbb{Z}$. Então

$$a^p \equiv a \bmod p. \tag{2.9}$$

Em particular, se p não divide a ($p \nmid a$), então

$$a^{p-1} \equiv 1 \bmod p. \tag{2.10}$$

Teorema de Euler: Se a e n são inteiros tais que $\text{mdc}(a,n) = 1$ então

$$a^{\varphi(n)} \equiv 1 \bmod n \tag{2.11}$$

Estes teoremas encontram-se enunciados em [14].

Seja $\text{mdc}(a,n) = 1$ e $\text{ord}_n(a) = \varphi(n)$. Então a é uma *raiz primitiva* de n. consideremos o caso particular em que p é um primo. Uma *raiz primitiva* de p é um inteiro g tal que $\text{ord}_p(g) = p-1$ (ver [15, 16]). Este resultado é um caso especial do primeiro pois sendo p um primo $\varphi(p) = p-1$.

2.1.5 Exponenciação modular rápida

Consideremos as seguintes duas propriedades da multiplicação e exponenciação modular:

$$mn \bmod c = [(m \bmod c) \times (n \bmod c)] \bmod c \tag{2.12}$$

e

$$a^b \bmod c = ((a \bmod c)^b) \bmod c. \tag{2.13}$$

Podemos facilmente notar que a^b torna-se rapidamente muito grande até para valores de b ainda pequenos, o que torna o algoritmo de exponenciação normal um processo muito lento ou mesmo praticamente impossível quando se utilizam valores grandes de b.

Uma forma de tornar esta exponenciação muito mais rápida, é utilizar o algoritmo de exponenciação modular rápida em vez do normal. Este, começa por decompor b na forma binária (base 2). Deste modo teremos $b = (b_{k-1}\ldots b_1 b_0)_2$, onde k é o número de bits de b. Assim podemos pegar nas equações (2.12) e (2.13) e reescrever

$$\begin{aligned} a^b \bmod c &= a^{b_0 + b_1 2 + \ldots + b_{k-1}2^{k-1}} \bmod c \\ &= [(a^{b_0} \bmod c) \times (a^{b_1 2} \bmod c) \times \ldots \times (a^{b_{k-1}2^{k-1}} \bmod c)] \bmod c \\ &= \left[\prod_{i=0}^{k-1} a^{b_i 2^i} \bmod c\right] \bmod c. \end{aligned} \tag{2.14}$$

2.2 Criptografia pré-quântica

Um criptossistema que usa uma série de operações, E_e, para cifrar e uma série de operações, D_d, para decifrar é considerado um criptossistema de chave-pública, PKC, se, para cada par de chaves (e,d), a chave de cifrar e é pública enquanto que a chave para decifrar d é mantida privada. Este criptossistema também precisa de satisfazer a condição de que é impossível em tempo polinomial descobrir d a partir de e (ver [17]).

Nesta secção iremos explicar, de forma breve, as principais diferenças entre problemas do tipo P e problemas do tipo NP para depois descrevermos o criptossistema de chave pública mais utilizado atualmente, o RSA, que apresenta um problema de fatorização considerado NP-Difícil.

2.2.1 P versus NP

Quanto tempo é necessário para executar computacionalmente um algoritmo? A resposta da ciência computacional a esta questão, não é em unidades de tempo, mas sim relativa ao número de elementos que o algoritmo terá que manipular durante todo o processo de execução.

Imaginemos que temos um vetor. Se nós quisermos escrever um algoritmo simples que nos diga qual o tamanho desse vetor, facilmente constatamos que, para este exemplo, o tempo de execução desse algoritmo é diretamente proporcional ao número de operações por ele executadas que por sua vez é diretamente proporcional ao número de elementos do vetor que designaremos por N. No entanto, outros algoritmos podem possuir tempos de execução proporcionais a N^2, N^3 ou $N \times \log N$, entre outros. A uma expressão matemática que envolva N's ou potências de base N, designamos por polinomial que é o que o P significa. Assim, à classe de problemas que podem ser resolvidos em tempo polinomial, denominamos por problemas do tipo P (ver [18]). Se a prova a uma solução de um problema poder ser verificada em tempo polinomial, dizemos que esse problema está na *classe NP*. Aos problemas mais difíceis dessa classe, designamos por *NP-Difícil*.

2.2.2 RSA

O RSA foi inventado em 1977, no MIT, por Ron Rivest, Adi Shamir e Leonard Adleman [14]. Este foi um dos primeiros PKC a surgir e é considerado um dos criptossistemas mais seguros, pois, até hoje todos os ataques clássicos construídos contra ele não podem ser executados em tempo polinomial. No entanto, o seu algoritmo é relativamente lento. Assim, o RSA é frequentemente utilizado na transmissão segura de chaves, em vez de ser utilizado para, por exemplo, cifrar dados ou mensagens.

Vamos supor que o Alice quer cifrar uma mensagem usando o RSA e enviar para o Bob. Para uma melhor explicação do algoritmo, este vai ser dividido em duas partes:

1. **Geração das chaves**

   (a) Bob gera dois números primos aleatórios, p e q, grandes e com o mesmo número de algarismos ou com um número de algarismos muito próximo (por exemplo, 300 dígitos cada) (ver [17]). A diferença $p - q$ deve ser grande para evitar que n possa ser fatorizado através de alguns ataques.

   (b) Calcula $n = pq$ e $\varphi(n) = (p-1)(q-1)$.

   (c) Escolhe aleatoriamente um inteiro e tal que $1 < e < \varphi(n)$ e $\text{mdc}(e, \varphi(n)) = 1$.

   (d) Calcula o único inteiro d tal que $1 < d < \varphi(n)$ e $ed \equiv 1 \mod \varphi(n)$.

   (e) Publica (n, e) como sendo a sua chave pública. Mantém secreto $(d, p, q, \varphi(n))$ como sendo a sua chave privada.

2. **Cifrar e decifrar**

   O Alice tem uma mensagem u para cifrar. Para simplificar vamos supor que u só pode ter no máximo $N = 27$ símbolos distintos que são, por exemplo, as 26 letras do alfabeto juntamente com o espaço. Faz-se corresponder a cada um desses símbolos, de forma ordenada, um número inteiro de 0 a 26.

   Cifrar

   Alice pega na chave pública do Bob.

   (a) Calcula k tal que $N^k < n < N^{k+1}$. Este valor de k corresponderá ao número de símbolos que ele irá cifrar a cada vez.

(b) Divide u em x blocos m_j, de k símbolos, onde $j = 0, 1, \ldots, x-1$. Se m_{x-1} não tiver k caracteres, acrescenta espaços à direita até este atingir o tamanho k.

(c) Faz corresponder cada ao i-ésimo símbolo de m_j o respetivo número n_i (por exemplo: $P \to 16$).

(d) Calcula, para cada m_j, o correspondente $M_j = (n_0, n_1, \ldots, n_{k-1})_{27} = n_0 \times N^0 + n_2 \times N^1 + \ldots + n_{k-1} \times N^{k-1}$.

(e) Computa $C_j = {M_j}^e \bmod n$.

(f) Envia C_j.

Decifrar

O Bob quer decifrar a mensagem cifrada C_j enviada pela Alice. Note-se que, Bob também sabe o valor de k e também sabe qual a correspondência entre cada símbolo e cada número (0 a 26).

(a) Computa $M_j = {C_j}^d \bmod n$.

(b) Depois de obter todos os M_j, escreve $M_j = (n_0, n_2, \ldots, n_{k-1})_{27}$

(c) Substitui cada carater de n_j pelo respetivo símbolo, obtendo assim m_j.

(d) Por último, faz a concatenação ordenada dos vários m_j e obtém a mensagem original, u.

Este algoritmo pode ser visto como uma função injetiva, pois, a cada bloco x irá corresponder um único C_j. A descodificação também é única, ou seja, a cada C_j corresponde um único bloco x. Assim mensagem original u será sempre recuperada.

A segurança da maioria dos PKC clássicos, reside no facto de ainda não existir um algoritmo de fatorização polinomial clássico. Pegando, por exemplo, no RSA, para decifrar uma mensagem por ele cifrada, precisaríamos de saber o d que só conseguiríamos descobrir se conhecêssemos os fatores de N. No entanto, com o surgimento e evolução da computação quântica, nomeadamente, o aparecimento do algoritmo de Shor, este problema de fatorização, antes considerado NP-Difícil, tornou-se num problema fácil, pois, pode ser resolvido em tempo polinomial num computador quântico. Assim, no próximo capítulo iremos estudar e implementar o algoritmo de Shor quântico.

Capítulo 3

Computação Quântica

Not only is the Universe stranger than we think, it is stranger than we can think.

Werner Heisenberg

O estudo da computação quântica iniciou-se em meados da década de 50, mantendo-se apenas teórico até 1982, quando foi apresentada uma proposta de utilização de sistemas quânticos na construção de computadores. Esta foi dada por Richard Feynman após referenciar as dificuldades em simular sistemas quânticos em computadores clássicos (ver [19]). No entanto, só na altura em que o algoritmo de Shor foi descoberto é que se deu mais importância a esta ideia e um novo alento à computação quântica.

Ao longo deste capítulo iremos abordar alguns conceitos da física e da computação quântica. Iremos estudar e implementar o algoritmo de Shor num simulador clássico de um sistema quântico e concluir sobre as limitações práticas que este algoritmo apresenta quando implementado num computador quântico real.

3.1 Fundamentos da Física Quântica

A física quântica é um ramo da física que estuda os fenómenos que acontecem a nível atómico e subatómico, ou radiação de muito baixa intensidade (associada a um número baixo de fotões). Esta, inicialmente conhecida como mecânica quântica, surgiu nos primeiros anos do século XX com Max Planck como pioneiro no desenvolvimento dos seus conceitos básicos. Desde então, esta área sofreu uma enorme evolução tornando-se a base de vários ramos da física e da química. Nesta secção iremos explicar e enunciar alguns fundamentos base da física quântica.

3.1.1 Produto interno e espaço de Hilbert

Antes de introduzirmos o conceito de *produto interno*, vamos introduzir a notação que será utilizada ao longo deste capítulo denotada por *notação de Dirac* ou *notação bra-ket*. Nesta notação usa-se parêntesis do tipo $\langle\rangle$ e uma barra vertical. Nesta notação um vetor (*ket*) é denotado por $|\psi\rangle$ e o seu hermítico conjugado é um vetor *bra* denotado por $\langle\psi|$. Por exemplo, seja $|\psi\rangle = \begin{bmatrix} z_1 & z_2 & z_3 \end{bmatrix}^T$, um vetor em $\mathbb{C}^3$, então $\langle\psi| = \begin{bmatrix} z_1^* & z_2^* & z_3^* \end{bmatrix}$.

Nesta notação, o produto interno entre dois estados, $|\psi\rangle$ e $|\phi\rangle$, é representado por $\langle\psi|\phi\rangle$ ou pelo seu complexo conjugado $\langle\phi|\psi\rangle$ que designaremos por *braket*.

Os estados quânticos de dimensão n são representados por vetores ou kets $|\psi\rangle$ de um espaço vetorial $\mathbb{C}^n$ sobre $\mathbb{C}$, onde está definido um produto interno com as seguintes regras:

- $\langle\psi|\psi\rangle \geq 0$,
- $\langle\psi|\phi\rangle = (\langle\phi|\psi\rangle)^*$,
- $(a\langle\psi| + b\langle\phi|)|\psi\rangle = a\langle\psi|\psi\rangle + b\langle\phi|\psi\rangle$,

onde a e b são complexos [2, 20]. A norma de um vetor $|\psi\rangle$ pode ser definida como

$$\|\psi\| = \sqrt{\langle\psi|\psi\rangle}. \tag{3.1}$$

Assim, se

$$\langle\psi|\phi\rangle = 0, \tag{3.2}$$

os estados $\langle\psi|$ e $|\phi\rangle$ são ortogonais entre si. O produto interno de um estado consigo próprio é sempre um número real não negativo. Se

$$\langle\psi|\psi\rangle = 1, \tag{3.3}$$

diz-se que o estado $|\psi\rangle$ está normalizado. A este tipo de espaços vetoriais dá-se o nome de *espaço de Hilbert* denotado por $\mathscr{H}$.

3.1.2 Postulados

Nesta secção iremos descrever, de forma breve, alguns postulados da mecânica quântica, essenciais e suficientes para compreender o conteúdo deste capítulo. Estes postulados foram ordenados e descritos de acordo com [19–22].

- **1º Postulado: Espaço de estados -** O estado de qualquer sistema físico pode ser representado por um vetor unitário $|\phi\rangle \in \mathscr{H}$.

  Se existirem $|\phi_1\rangle$ e $|\phi_2\rangle$, possíveis estados do sistema, então a sobreposição

  $$|\phi\rangle = a\,|\phi_1\rangle + b\,|\phi_2\rangle \tag{3.4}$$

  é também um estado do sistema, onde a e b são números complexos tais que $|a|^2 + |b|^2 = 1$.

- **2º Postulado: Evolução -** A evolução de um sistema físico fechado é descrito por uma transformação unitária, ou seja, o estado $|\phi(t_1)\rangle$ do sistema está relacionado com o estado $|\phi(t_2)\rangle$ por um operador unitário $\hat{U}$ que depende apenas dos instantes de tempo tal que,

  $$|\phi(t_2)\rangle = \hat{U}\,|\phi(t_1)\rangle\,. \tag{3.5}$$

- **3º Postulado: Medição -** Uma medição quântica é descrita por uma série de operadores de medição $\{M_m\}$ que atuam no estado do sistema a ser medido, onde o índice m denota um resultado pontual para a medição. Assim, se o estado do sistema imediatamente antes da medição for $|\phi\rangle$, então a probabilidade de ocorrer o resultado m é dada por

  $$p(m) = \langle\phi|\,M_m^\dagger M_m\,|\phi\rangle\,, \tag{3.6}$$

  e o estado do sistema após uma medição com resultado m é

  $$\frac{M_m\,|\phi\rangle}{\sqrt{p(m)}} = \frac{M_m\,|\phi\rangle}{\sqrt{\langle\phi|\,M_m^\dagger M_m\,|\phi\rangle}}. \tag{3.7}$$

  Este efeito é conhecido como o colapso do estado do sistema, que se refere ao facto do estado final do sistema depender do resultado da medição. Os operadores de medição satisfazem a equação

  $$\sum_m M_m^\dagger M_m = I, \tag{3.8}$$

  que se traduz no facto de que a soma das probabilidades de todos os resultados é igual a 1, ou seja,

  $$\sum_m p(m) = \sum_m \langle\phi|\,M_m^\dagger M_m\,|\phi\rangle = 1. \tag{3.9}$$

  Assim, a previsão do resultado na medição quântica é de natureza probabilística.

- **4º Postulado: Sistemas compostos -** O espaço de estados de um sistema físico composto é o produto tensorial dos espaços dos sistemas físicos que o compõe. Assim, se tivermos sistemas numerados de $1, 2, \ldots, n$, o estado do sistema composto por estes é dado por

$$\begin{aligned} |\phi\rangle &= |\phi_1\rangle \otimes |\phi_2\rangle \otimes \ldots \otimes |\phi_n\rangle \\ &= |\phi_1\rangle\, |\phi_2\rangle \ldots |\phi_n\rangle \\ &= |\phi_1, \ldots, \phi_2, \phi_n\rangle\,. \end{aligned} \tag{3.10}$$

  A dimensão do espaço composto é o produto das dimensões dos espaços que o constituem.

3.1.3 Princípio da Não Clonagem

O termo *Não Clonagem* é principalmente utilizado em informação quântica para afirmar que estados quânticos arbitrários não podem ser copiados de forma exata. De facto, este afirma que num espaço de Hilbert $\mathscr{H}$, não existe qualquer transformação unitária $U : \mathscr{H} \otimes \mathscr{H} \longmapsto \mathscr{H} \otimes \mathscr{H}$, tal que (ver [22])

$$U(|\psi\rangle\, |s\rangle) = |\psi\rangle\, |\psi\rangle\,. \tag{3.11}$$

Podemos facilmente provar este teorema por redução ao absurdo. Para isso, vamos supor que temos três estados quânticos, $|\psi_1\rangle$, $|\psi_2\rangle$ e $|s\rangle$, e que temos essa transformação unitária U, capaz de clonar qualquer estado quântico $|\psi\rangle$, contido num primeiro registo, para um segundo registo que inicialmente contém $|s\rangle$, ou seja,

$$U(|\psi\rangle\, |s\rangle) = |\psi\rangle\, |\psi\rangle\,. \tag{3.12}$$

Sejam $|\psi_1\rangle$ e $|\psi_2\rangle$ ortogonais. Ao aplicarmos U em cada um deles ficaríamos com

$$U(|\psi_1\rangle\, |s\rangle) = |\psi_1\rangle\, |\psi_1\rangle \tag{3.13}$$

e

$$U(|\psi_2\rangle\, |s\rangle) = |\psi_2\rangle\, |\psi_2\rangle\,. \tag{3.14}$$

Consideremos agora um outro estado $|\psi_3\rangle = \frac{|\psi_1\rangle + |\psi_2\rangle}{\sqrt{2}}$. Então, por linearidade,

$$\begin{aligned} U(|\psi_3\rangle\, |s\rangle) &= \frac{U(|\psi_1\rangle\, |s\rangle) + U(|\psi_2\rangle\, |s\rangle)}{\sqrt{2}} \\ &= \frac{|\psi_1\rangle\, |\psi_1\rangle + |\psi_2\rangle\, |\psi_2\rangle}{\sqrt{2}}. \end{aligned} \tag{3.15}$$

No entanto, se U é uma transformação de clonagem, então

$$\begin{aligned} U(|\psi_3\rangle\, |s\rangle) &= |\psi_3\rangle\, |\psi_3\rangle \\ &= \frac{|\psi_1\rangle\, |\psi_1\rangle + |\psi_1\rangle\, |\psi_2\rangle + |\psi_2\rangle\, |\psi_1\rangle + |\psi_2\rangle\, |\psi_2\rangle}{2}. \end{aligned} \tag{3.16}$$

Por inspeção, podemos verificar que o resultado em 3.15 é diferente do resultado em 3.16. Assim, como queríamos demonstrar, não existe qualquer transformação unitária U capaz de clonar um estado quântico arbitrário.

3.2 Fundamentos da Computação Quântica

Nesta secção vamos introduzir alguns conceitos base, essenciais para a compreensão de alguns conteúdos que iremos posteriormente abordar.

3.2.1 Bits e Qubits

A informação digital baseia-se na diferença, distinção, ou discriminação, entre dois estados possíveis. Nos sistemas clássicos, a informação é representada através de bits, dígitos binários, que podem assumir o valor de 0 ou 1 correspondendo cada um deles a um desses estados [23].

Nos sistemas quânticos, a informação é representada por qubits (bits quânticos). Estes são a unidade básica de informação processada pelos computadores quânticos e podem assumir valores de 0 ou 1, que serão designados por estados $|0\rangle$ e $|1\rangle$, respetivamente, mas também uma sobreposição destes dois. Deste modo, um vetor de estado arbitrário que representará o qubit pode ser escrito na forma

$$|\psi\rangle = a\,|0\rangle + b\,|1\rangle \tag{3.17}$$

onde a e b são números complexos tais que, de modo verificar a condição de normalização para vetores de estado, $|a|^2 + |b|^2 = 1$ (ver [19]). Estamos num espaço a duas dimensões cuja base pode ser $\{|0\rangle, |1\rangle\}$, designada por base computacional. Assim, na forma vetorial cada um destes estados, $|0\rangle$ e $|1\rangle$, pode ser descrito como $|0\rangle = \begin{bmatrix}1\\0\end{bmatrix}$ e $|1\rangle = \begin{bmatrix}0\\1\end{bmatrix}$, ficando assim

$$|\psi\rangle = a\begin{bmatrix}1\\0\end{bmatrix} + b\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}a\\b\end{bmatrix}. \tag{3.18}$$

Os valores $|a|^2$ e $|b|^2$ correspondem à probabilidade de, no momento da medição, este colapsar para o estado $|0\rangle$ ou $|1\rangle$, respetivamente.

Para um multiqubit composto por n qubits, a dimensão do espaço é 2^n. Logo, o estado do multiqubit é

$$|\psi\rangle = \begin{bmatrix}\psi_1\\ \psi_2\\ \vdots\\ \psi_{2^n}\end{bmatrix}. \tag{3.19}$$

Quando acharmos necessário iremos utilizar a notação $|\psi\rangle_n$ para indicar que um multiqubit, ou um registo, é constituído por n qubits.

3.2.2 Esfera de Bloch

Uma forma simples de visualizar o comportamento de um qubit é descrevê-lo usando a Esfera de Bloch (figura 3.1). Pegando no vetor estado dado pela equação 3.17, como a e b são complexos, podemos decompor cada componente a e b em coordenadas polares escrevendo $a = r_1 e^{i\theta_1}$ e $b = r_2 e^{i\theta_2}$, onde r_1 e r_2 são reais e $r_1^2 + r_2^2 = 1$. Assim podemos reescrever o vetor estado do qubit como

$$\begin{aligned}|\psi\rangle &= r_1 e^{i\theta_1}|0\rangle + r_2 e^{i\theta_2}|1\rangle \\ &= e^{i\theta_1}(r_1|0\rangle + r_2 e^{i(\theta_2-\theta_1)}|1\rangle).\end{aligned} \tag{3.20}$$

A fase global, θ_1 em $e^{i\theta_1}$, é irrelevante pois não afeta a medição (ver [22, 24]), assim podemos reescrever

$$|\psi\rangle = r_1|0\rangle + r_2 e^{i(\theta_2-\theta_1)}|1\rangle. \tag{3.21}$$

Agora, se voltarmos a alterar os parâmetros r_1 e r_2 para $r_1 = \cos\frac{\theta}{2}$ e $r_2 = \sin\frac{\theta}{2}$ e fizermos $\phi = \theta_2 - \theta_1$, o vetor de estado fica

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + \sin\frac{\theta}{2}e^{i\phi}|1\rangle. \tag{3.22}$$

Assim, o estado $|\psi\rangle$ de um qubit pode ser representado como um ponto à superfície duma esfera unitária (esfera de Bloch) ou, de forma equivalente, através do vetor que liga o centro da esfera a um ponto da superfície (vetor de Bloch) onde, θ e ϕ são os ângulos das coordenadas esféricas. Nesta esfera, $|0\rangle$ e $|1\rangle$ encontram-se no eixo-Z onde $|0\rangle$ é corresponde ao vetor $(0,0,1)$ e $|1\rangle$ ao vetor $(0,0,-1)$.

3.2.3 Portas lógicas quânticas

Chamaremos *porta lógica quântica* a qualquer sistema quântico capaz de aplicar uma operação unitária em um ou mais qubits [23]. Analogamente podemos definir circuitos quânticos como sistemas constituídos por estas portas e que cujos passos computacionais estão sincronizados no tempo. O tamanho de cada circuito é dado pelo número de portas que o constitui [25]. Estas portas quânticas

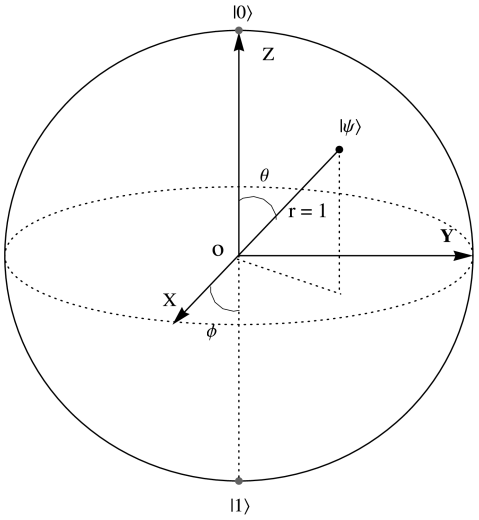


Figura 3.1: Representação do estado de um qubit $|\psi\rangle$ como um ponto na superfície da Esfera de Bloch.

são operadores, reversíveis e unitários, que podem ser representados por matrizes $2^n \times 2^n$, onde n representa o número de qubits de entrada e saída em cada um deles.

Nesta secção apenas enunciaremos as portas lógicas que consideramos mais relevantes.

3.2.4 Portas lógicas quânticas de um único qubit

Na figura 3.2 temos a representação geral de uma porta quântica de um único qubit. Esta transforma o estado $|\psi_{\text{in}}\rangle$ num estado $|\psi_{\text{out}}\rangle = U|\psi_{\text{in}}\rangle$. Nesta e nas figuras seguintes, que representarão portas quânticas ou circuitos quânticos, cada linha horizontal representará um único qubit. Estes estarão ordenados do mais significativo para o menos significativo sendo o menos significativo, o de cima, e o mais significativo, o de baixo. A evolução temporal será da esquerda para a direita.

Por consequência do Principio da Não Clonagem, não podemos ter portas lógicas capazes de mapear todas as sobreposições quânticas, pois, após a primeira medição, obtém-se um valor clássico e o estado colapsa para o estado quântico correspondente, perdendo-se completamente toda a informação sobre o estado do sistema antes da medição. Assim, utilizam-se portas que copiam apenas bits clássicos. O símbolo correspondente à medição está representado na figura 3.3.

Figura 3.2: Forma geral de uma porta quântica para um único qubit.

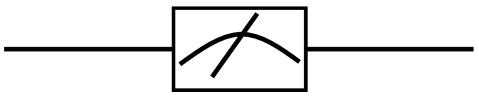


Figura 3.3: Símbolo correspondente à medição.

- **Portas lógicas de Pauli (*Pauli gates*):**

  As 3 portas de Pauli denominadas de X, Y e Z são as transformações mais básicas que se podem aplicar a um único qubit. A sua representação e respetivas matrizes podem ser observadas na figura 3.4.

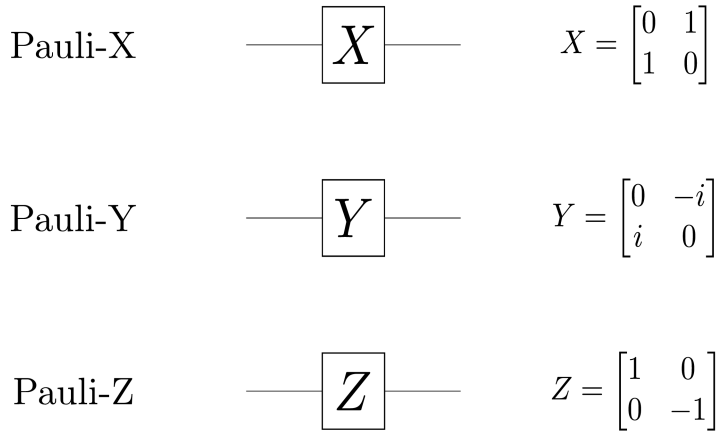


Figura 3.4: Portas lógicas de Pauli e respetiva representação simbólica e matriz.

Ao aplicar cada uma delas ao qubit $|\psi\rangle = a|0\rangle + b|1\rangle$ obtém-se:

- **Pauli-X:** $X(a|0\rangle + b|1\rangle) = b|0\rangle + a|1\rangle$
  Logo esta será a equivalente à port NOT da computação clássica, ou seja, troca $|0\rangle$ por $|1\rangle$ e vice versa.

- **Pauli-Y:** $Y(a|0\rangle + b|1\rangle) = -i(-b|0\rangle + a|1\rangle)$
  Esta transforma $|0\rangle$ em $-i|1\rangle$ e $|1\rangle$ em $i|0\rangle$.

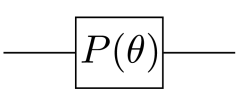
- **Pauli-Z:** $Z(a|0\rangle + b|1\rangle) = a|0\rangle - b|1\rangle)$
  Não afeta o estado $|0\rangle$. Transforma $|1\rangle$ em $-|1\rangle$.

- **Porta Hadamard:**
  A Hadamard (figura 3.5) é uma porta quântica muito importante. Esta transforma $|0\rangle$ em $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ e $|1\rangle$ em $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$, ou seja, transforma $|0\rangle$ e $|1\rangle$ numa sobreposição de $|0\rangle$ e $|1\rangle$ com igual peso. O mesmo acontece se aplicarmos uma Hadamard a cada qubit de um qubit múltiplo. Por exemplo, se aplicássemos Hadamards ao estado de dois qubits $|00\rangle$ iríamos obter $\frac{|00\rangle+|01\rangle+|10\rangle+|11\rangle}{\sqrt{2^2}}$ que na forma decimal corresponde aos valores 0, 1, 2 e 3 com igual probabilidade. Esta porta quântica é representada por H. No entanto, quando esta é aplicada num multiqubit, com por exemplo n qubits, designa-se por H^n.

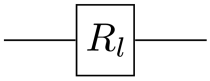
Figura 3.5: Porta quântica Hadamard e respetiva representação matricial.

- **Porta de fase (*Phase gate or phase shift gate*):**
  Como referido anteriormente, o estado de qualquer qubit pode ser representado como um ponto à superfície da Esfera de Bloch. A porta quântica de fase, $P(\theta)$, adiciona uma fase θ ao estado de um qubit que, na Esfera de Bloch, corresponde a uma rotação de θ em torno do eixo-z, ou seja, $P(\theta)(a|0\rangle + b|1\rangle) = a|0\rangle + be^{i\theta}|1\rangle$.

  Por vezes, ao longo deste capítulo, em vez da sua representação mais comum $P(\theta)$, esta porta será representada por R_l sendo $\theta = \frac{2\pi}{2^l}$, pois, facilitará a compreensão de alguns circuitos quânticos que se seguem. Ambas as representações simbólicas assim como as respetivas representações matriciais encontram-se na figura 3.6.



$$P(\theta) = \begin{bmatrix} 1 & 0 \\ 0 & \mathrm{e}^{i\theta} \end{bmatrix}$$



$$R_l = \begin{bmatrix} 1 & 0 \\ 0 & \mathrm{e}^{\frac{i2\pi}{2^l}} \end{bmatrix}$$

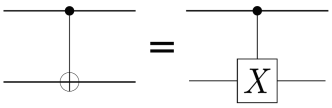
Figura 3.6: Porta quântica de fase (duas notações equivalentes).

Portas quânticas multiqubit

Qualquer porta multiqubit pode ser obtida utilizando apenas portas de um único qubit e portas CNOT (Controlled NOT gate). Ao longo desta secção enunciarei quais as portas multiqubit mais utilizadas e como estas podem ser obtidas usando apenas as portas CNOT e as de um único qubit.

- **Porta quântica CNOT:**

  É a mais básica, comum e das mais importantes portas quânticas. Utiliza apenas dois qubits, sendo um apenas de controlo e o outro o alvo. Se o qubit de controlo estiver no estado $|1\rangle$ esta aplica o operador Pauli-X (NOT) no bit alvo, caso contrário, não faz nada. É uma Pauli-X controlada. O seu circuito assim como a sua forma matricial está representada na figura 3.7. Nestes circuitos o qubit de controlo é representado como aquele que possui o simbolo $\bullet$ e o qubit alvo como aquele que possui, no caso do CNOT, o símbolo $\oplus$ ou o da porta X.

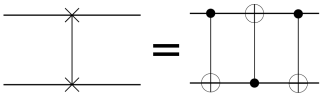


$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Figura 3.7: Porta quântica CNOT e respetiva representação matricial.

- **Porta quântica SWAP:**

  Tem a função de trocar o estado de dois qubits entre si, podendo ser construída apenas com portas CNOT (figura 3.8).



$$\text{SWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figura 3.8: Porta quântica SWAP, circuito equivalente com portas CNOT e respetiva representação matricial.

- **Portas quânticas controladas:**

  Consideremos U uma porta lógica quântica de um único qubit. Então, pode-se construir uma U-Controlada de dois qubits para que U opere no segundo qubit (alvo) se o primeiro qubit (controlo) estiver no estado $|1\rangle$.

  As portas lógicas quânticas controladas mais comuns são portas fase controladas. Uma fase controlada pode ser obtida conforme a representação da figura 3.9, onde R_l^* representa o hermítico conjugado de R_l.

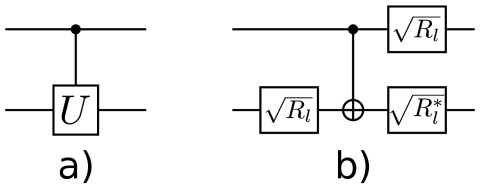
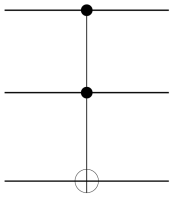


Figura 3.9: (a) Porta quântica arbitrária U-controlada. (b) Circuito equivalente para uma fase R_l-controlada.

- **Porta quântica Toffoli:**

  É a porta equivalente à porta AND da computação clássica. Precisa de três qubits para ser representada, dois de controlo e um alvo. Se os dois qubits de controlo se encontrarem no estado $|1\rangle$ ela aplica a operação NOT (Pauli-X) no qubit alvo, caso contrário não faz nada. A sua representação simbólica e matricial encontra-se na figura 3.10.



$$\text{Toffoli} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Figura 3.10: Porta quântica Toffoli e respetiva representação matricial.

3.2.5 Entrelaçamento quântico e paralelismo quântico

O entrelaçamento quântico é uma das propriedades da física quântica importante na diferenciação da computação quântica. Este é um fenómeno que ocorre quando dois ou mais sistemas estão correlacionados de tal modo que o estado de um não pode ser descrito de forma independente do estado dos outros. Por exemplo, vamos supor que temos dois eletrões entrelaçados com *spins* opostos. Então, mesmo que estes sejam separados por uma distância muito grande, sempre que efetuarmos uma medição do *spin* num deles, que poderá ser *up* ou *down* com 50% de probabilidade, e de seguida medirmos o do segundo, este será sempre o oposto do *spin* da medição do primeiro.

Este fenómeno pode ser facilmente demonstrado pelo resultado de um circuito simples como o da figura figura 3.11. O estado inicial é $|\psi\rangle = |00\rangle$. Ao aplicarmos inicialmente uma Hadamard no primeiro qubit obteremos nesse ponto do circuito $|\psi_1\rangle = \frac{|00\rangle+|10\rangle}{\sqrt{2}}$. No entanto, depois de aplicar uma CNOT no segundo qubit com controlo no primeiro, podemos perceber que obteremos $|\psi_2\rangle = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$, ou seja, o estado do segundo qubit está correlacionado com o primeiro de tal modo que, depois de efetuarmos a medição do estado do primeiro que será $|0\rangle$ ou $|1\rangle$ com 50% de probabilidade, saberemos com 100% de certeza o estado do segundo. Deste modo podemos dizer que estes dois qubits estão num estado entrelaçado.

Vamos agora supor que no mesmo circuito da figura 3.11, x representa o estado do primeiro qubit após a aplicação da Hadamard. Com o segundo qubit inicializado a $|0\rangle$, a operação CNOT pode ser entendida como $f(x) = x$. Deste modo, antes da medição, há uma sobreposição de 2^1 valores possíveis para x, $|0\rangle$ ou $|1\rangle$, logo, este segundo qubit contém simultaneamente a informação de $f(x)$ para todos esses possíveis valores de x. A este efeito é chamado de *paralelismo quântico*. Deste modo com n qubits conseguimos trabalhar simultaneamente com 2^n valores. Ou seja, enquanto um computador clássico apenas aplica uma função a uma determinada sequência de bits, devido a este paralelismo, um computador quântico consegue aplicar uma função a todas as realizações de um determinado número de qubits.

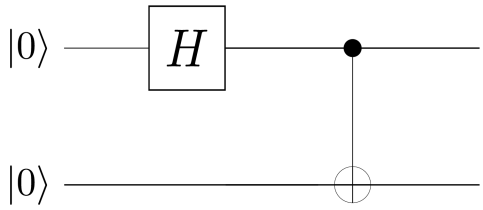


Figura 3.11: Exemplo de um circuito onde ocorre entrelaçamento quântico.

Além de todo este potencial, a computação quântica possui enormes limitações. Uma delas ocorre no instante da medição. Apesar de, devido ao paralelismo quântico, conseguirmos trabalhar com 2^n valores simultaneamente, estes não estão acessíveis de forma independente, ou seja, após cada medição iremos obter aleatoriamente, com maior ou menor probabilidade de acordo com a sobreposição em causa, um único estado entre todos os possíveis. Assim, a informação relevante que podemos extrair da computação quântica terá que ser obtida de forma indireta e, para isso, é necessário utilizar alguns algoritmos capazes de nos darem alguma informação sobre essa sobreposição. Um desses algoritmos é o algoritmo de Shor que descreveremos mais à frente e que se baseia na Transformada de Fourier Quântica, *Quantum Fourier Transform* (QFT).

3.3 Implementações práticas e decoerência

Computadores quânticos tem sido desenvolvidos com base em diferentes tipos de sistemas de qubits. Atualmente, após décadas de evolução, os sistemas mais promissores são baseados em *spins*, *circuitos supercondutores*, *armadilhas de iões* e *circuitos fotónicos* (ver [5]). Em todos estes sistemas, o qubit é um sistema físico de 2 níveis, um deles correspondendo ao $|0\rangle$ e o outro ao $|1\rangle$.

A construção de um computador quântico é muito complicada pois, a computação quântica requer que os sistemas de átomos ou fotões mantenham a coerência, isto é, mantenham a fase ao longo do tempo e que apenas evoluam de acordo com as portas aplicadas. Assim, os qubits precisam ser manipulados e ao mesmo tempo protegidos de fontes externas de calor ou radiação eletromagnética. No entanto, todos os qubits interagem com algo, com o sistema que os suporta, e sobre o qual nós não temos controlo, ou seja, não podemos medir as alterações ou aplicar portas quânticas de modo a obter algum resultado que nos permita reverter o efeito dessa interação. Apenas em alguns raros casos, esse efeito pode ser revertido pelo próprio sistema quântico, no entanto, nos restantes casos ou quando o experimento é de longa duração, aumentando assim o tempo de interação com o ambiente, ocorre a chamada *decoerência*. Assim, este termo, é geralmente utilizado para denotar a degradação irreversível de um estado quântico devido à sua interação com o ambiente externo [26]. Existem algumas formas de aumentar o tempo de coerência de um sistema quântico. O que geralmente se faz é baixar, dentro do possível, a temperatura do sistema e usar sistemas de controlo de temperatura pontuais de modo evitar a ocorrência de flutuações térmicas.

Outro grande problema que a computação quântica enfrenta, que reside já na parte da implementação das portas lógicas, é o facto de que, ao contrário do que foi feito no nosso simulador do computador quântico apresentado mais à frente, nem sempre é possível a ligação entre quaisquer qubits, sendo possível apenas o uso de portas lógicas multiqubit entre qubits vizinhos/ligados diretamente. Um exemplo disso pode ser observado na figura 3.12 onde se encontra o esquema de disposição dos qubits assim como as interações permitidas para portas lógicas de dois qubits, do computador quântico *IBM Q 5 Tenerife* do IBM, que usa um sistema de qubits baseado em supercondutores.

3.4 Circuitos quânticos

Como a computação quântica ainda está no nível de desenvolvimento de *hardware*, os algoritmos são ainda pensados como uma sequência de portas lógicas. Como não existe nenhuma porta lógica que nos dê diretamente a ordem de um número módulo N, a exponenciação modular nem mesmo uma

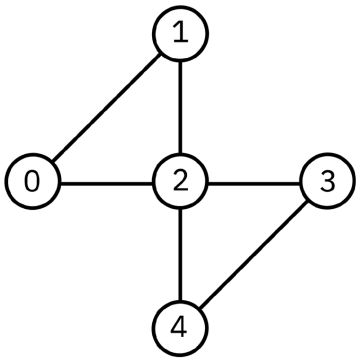


Figura 3.12: Esquema da disposição dos qubits no computador quântico, *IBM Q 5 Tenerife*, do IBM.

soma simples de inteiros $a + b$, antes de implementarmos o algoritmo de Shor, teremos que aprender a implementar operações mais básicas que depois serão utilizadas para construção de operações mais complexas até chegarmos ao circuito que corresponde à parte quântica do algoritmo de Shor. Essas operações mais básicas são a adição e multiplicação quântica. No entanto, para implementar estas operações, precisamos ainda de aprender sobre o funcionamento e implementação de uma ferramenta muito útil na computação quântica. Essa ferramenta é a transformada de Fourier quântica.

3.4.1 Transformada de Fourier Quântica

A Transformada de Fourier Quântica, QFT, permite-nos fazer análise espetral como por exemplo obter períodos de funções, à semelhança do que acontece com a transformada de Fourier clássica, mas também nos permite realizar operações aritméticas de forma mais eficiente, utilizando menos qubits, num computador quântico. Antes de definirmos a QFT vamos primeiro definir a Transformada de Fourier clássica de uma função $f(t)$ como sendo

$$F(w) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) \mathrm{e}^{iwt} dt \tag{3.23}$$

e a sua inversa como

$$f(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} F(w) \mathrm{e}^{-iwt} dt. \tag{3.24}$$

Do mesmo modo podemos também definir a Transformada de Fourier Discreta, *Discrete Fourier Transform* (DFT), que não é mais que uma Transformada de Fourier de uma série de dados discretos. Para isso vamos supor que temos uma sequência de N números complexos $x_0, x_1, \ldots, x_{N-1}$. Então a DFT desta sequência é

$$X_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_n \mathrm{e}^{i \frac{2\pi nk}{N}}, \tag{3.25}$$

onde k varia de 0 até $N-1$. Assim podemos verificar que a DFT transforma uma sequência de dados discretos numa outra sequência de dados discretos. Analogamente podemos definir a sua inversa como

$$x_n = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} X_k \mathrm{e}^{-i \frac{2\pi nk}{N}}. \tag{3.26}$$

Agora vamos supor que em vez de uma série de dados temos um estado quântico, definido num espaço de Hilbert de dimensão N e escrito na base $\{|n\rangle\}$, escrito na forma $|\psi\rangle = \sum_{n=0}^{N-1} a_n |n\rangle$. Nesta base o estado é descrito pelo vetor $\begin{bmatrix} a_0 & a_1 & \cdots & a_{N-1} \end{bmatrix}^T$ que não é mais que uma série de números complexos a_i tais que $\sum_i |a_i|^2 = 1$. Notemos que, este estado pode ser escrito com $n = \log_2(N)$ qubits. Uma QFT é uma operação de computação quântica que transforma esse estado quântico num

outro $|\psi'\rangle = \sum_{k=0}^{N-1} b_k |k\rangle$ onde b_k é a DFT de a_n. A QFT aplicada a cada um dos vetores da base $|n\rangle$ devolve

$$\mathrm{QFT}\,|n\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \mathrm{e}^{i\frac{2\pi nk}{N}} |k\rangle\,. \tag{3.27}$$

Assim a QFT permite-nos transformar um estado da base, $|n\rangle$, numa sobreposição de estados de base $|k\rangle$ cujos coeficientes são uma soma de fases $\mathrm{e}^{i\frac{2\pi nk}{N}}$ (ver [22]). Para nos ajudar a perceber o seu circuito quântico (figura 3.13) e algumas operações utilizadas mais à frente nesta dissertação, onde esta será aplicada, vamos reescrever 3.27 como

$$\frac{1}{\sqrt{2}}\left(|0\rangle + \mathrm{e}^{\frac{i\pi n}{2^0}}|1\rangle\right) \otimes \frac{1}{\sqrt{2}}\left(|0\rangle + \mathrm{e}^{\frac{i\pi n}{2^1}}|1\rangle\right) \otimes \ldots \otimes \frac{1}{\sqrt{2}}\left(|0\rangle + \mathrm{e}^{\frac{i\pi n}{2^{N-2}}}|1\rangle\right) \otimes \frac{1}{\sqrt{2}}\left(|0\rangle + \mathrm{e}^{\frac{i\pi n}{2^{N-1}}}|1\rangle\right). \tag{3.28}$$

A respetiva inversa (QFT^{-1}) da QFT em 3.27 aplicada a cada um dos vetores da base $|k\rangle$ devolve

$$\mathrm{QFT}^{-1}\,|k\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \mathrm{e}^{-i\frac{2\pi nk}{N}} |n\rangle \tag{3.29}$$

tal que

$$\mathrm{QFT}^{-1}\mathrm{QFT}\,|n\rangle = |n\rangle\,. \tag{3.30}$$

A figura 3.13 mostra o circuito quântico geral para uma QFT onde $|\phi(a)\rangle$ representa o estado $\mathrm{QFT}|a\rangle$. O circuito da QFT^{-1} pode ser obtido invertendo o circuito da direita para a esquerda e conjugando todas as fases.

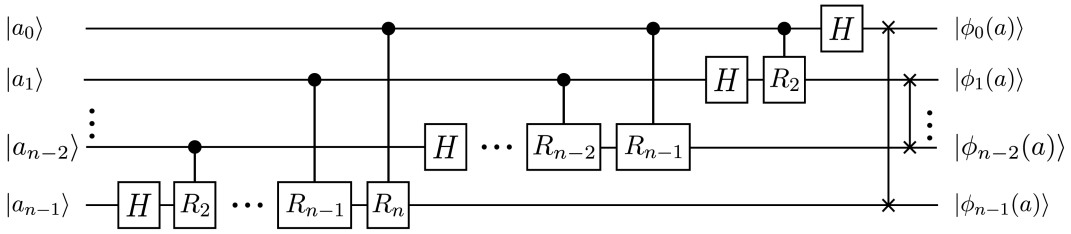


Figura 3.13: Representação geral do circuito da Transformada de Fourier Quântica.

À medida que l aumenta, a fase adicionada por R_l torna-se cada vez menor. Daí tornar-se pertinente a questão de como seria fisicamente possível, para valores grandes de l, aplicar diferenças de fase tão baixas? Para valores grandes de l, facilmente percebemos que a matriz R_l aproxima-se a uma matriz identidade. Assim, está provado em [27] que em caso de decoerência, que é experimentalmente inevitável devido ao tempo de interação do sistema quântico com o ambiente enquanto uma QFT de elevada ordem é executada, é melhor utilizar uma QFT aproximada que despreza essas fases muito baixas. Esta aproximação resolve o problema de não termos a precisão necessária para aplicar essas fases, logo estaríamos a aplicar fases erradas, e reduz o número de operações necessárias para uma QFT de n qubits, de $\frac{n(n+1)}{2}$ para $\frac{(2n-\log_2 n)(\log_2 n-1)}{2} \approx n\log_2 n$ operações (ver [28]), diminuindo assim o tempo de todo o experimento o que implica uma menor perda de coerência.

3.4.2 Adição e multiplicação quântica

Nesta subsecção descreveremos as operações aritméticas, adição e multiplicação modulares, necessárias para a exponenciação modular do algoritmo de Shor.

A primeira operação quântica que começaremos por descrever é a adição. Neste trabalho, vamos usar um algoritmo de adição desenvolvido em [28] e conhecido como *adição Draper*. Este algoritmo foi desenvolvido de forma a reduzir o número de qubits necessários [28, 29]. A adição é feita no domínio de Fourier aplicando uma sequência de fases controladas, que são mutuamente comutativas. Vamos supor que temos um registo quântico $|b'\rangle = |b\rangle_n |0\rangle_1$ cuja dimensão do espaço é $2^n \times 2 = 2^{n+1} = M$.

Então a QFT de $|b'\rangle$ será dada por

$$\begin{aligned}\mathrm{QFT}\,|b'\rangle &= \frac{1}{\sqrt{M}}\sum_{k=0}^{M-1} \mathrm{e}^{\frac{i2\pi b' k}{M}}\,|k\rangle \\ &= \frac{1}{\sqrt{2}}\left(|0\rangle + \mathrm{e}^{\frac{i\pi b'}{2^0}}\,|1\rangle\right)\otimes\frac{1}{\sqrt{2}}\left(|0\rangle + \mathrm{e}^{\frac{i\pi b'}{2^1}}\,|1\rangle\right)\otimes\ldots\otimes\frac{1}{\sqrt{2}}\left(|0\rangle + \mathrm{e}^{\frac{i\pi b'}{2^n}}\,|1\rangle\right). \quad (3.31)\end{aligned}$$

Notemos que foi adicionado um qubit extra, inicializado como $|0\rangle$, na parte da QFT. O motivo é que para representarmos na forma binária o resultado da adição de dois números de comprimento n, necessitamos de $n+1$ bits. Vamos agora adicionar $|a\rangle_n$ ao registo anterior já no domínio de Fourier. Então para isso basta substituir o b' da equação anterior por $(a+b')$, pois, se pensarmos um pouco, o resultado teria que ser igual a aplicar logo a QFT ao estado $|a+b'\rangle$. Assim, ficaremos com

$$\mathrm{QFT}\,|a+b'\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + \mathrm{e}^{\frac{i\pi(a+b')}{2^0}}\,|1\rangle\right)\otimes\frac{1}{\sqrt{2}}\left(|0\rangle + \mathrm{e}^{\frac{i\pi(a+b')}{2^1}}\,|1\rangle\right)\otimes\ldots\otimes\frac{1}{\sqrt{2}}\left(|0\rangle + \mathrm{e}^{\frac{i\pi(a+b')}{2^n}}\,|1\rangle\right). \quad (3.32)$$

Ao escrever a em binário, pois, é este que está a ser adicionado ao registo que contém a $\mathrm{QFT}\,|b'\rangle$, ficamos com

$$a = a_{n-1}2^{n-1} + a_{n-2}2^{n-2} + \ldots + a_1 2^1 + a_0 2^0. \quad (3.33)$$

Deste modo, no qubit menos significativo do registo da soma teremos

$$\frac{1}{\sqrt{2}}\left(|0\rangle + \mathrm{e}^{\left(\frac{i\pi(b' + a_{n-1}2^{n-1} + a_{n-2}2^{n-2} + \ldots + a_1 2^1 + a_0 2^0)}{2^n}\right)}\,|1\rangle\right) \quad (3.34)$$

que se rearranjarmos fica

$$\frac{1}{\sqrt{2}}\left(|0\rangle + \mathrm{e}^{\left(\underbrace{\frac{i\pi b'}{2^n}}_{\mathrm{QFT}|b'\rangle} + \underbrace{\frac{i\pi a_{n-1}}{2}}_{R_2\ \mathrm{ctrl}\ a_{n-1}} + \underbrace{\frac{i\pi a_{n-2}}{2^2}}_{R_3\ \mathrm{ctrl}\ a_{n-2}} + \ldots + \underbrace{\frac{i\pi a_1}{2^{n-1}}}_{R_n\ \mathrm{ctrl}\ a_1} + \underbrace{\frac{i\pi a_0}{2^n}}_{R_{n+1}\ \mathrm{ctrl}\ a_0}\right)}\,|1\rangle\right). \quad (3.35)$$

Do mesmo modo, para o segundo qubit menos significativo do registo da soma teremos

$$\frac{1}{\sqrt{2}}\left(|0\rangle + \mathrm{e}^{\left(\frac{i\pi(b' + a_{n-1}2^{n-1} + a_{n-2}2^{n-2} + \ldots + a_1 2^1 + a_0 2^0)}{2^{n-1}}\right)}\,|1\rangle\right) \quad (3.36)$$

que podemos escrever como

$$\frac{1}{\sqrt{2}}\left(|0\rangle + \mathrm{e}^{\left(\underbrace{\frac{i\pi b'}{2^{n-1}}}_{\mathrm{QFT}|b'\rangle} + \underbrace{\frac{i\pi a_{n-1}}{2^0}}_{R_1\ \mathrm{ctrl}\ a_{n-1}} + \underbrace{\frac{i\pi a_{n-2}}{2^1}}_{R_2\ \mathrm{ctrl}\ a_{n-2}} + \ldots + \underbrace{\frac{i\pi a_1}{2^{n-2}}}_{R_{n-1}\ \mathrm{ctrl}\ a_1} + \underbrace{\frac{i\pi a_0}{2^{n-1}}}_{R_n\ \mathrm{ctrl}\ a_0}\right)}\,|1\rangle\right). \quad (3.37)$$

Continuando, para o terceiro qubit menos significativo temos

$$\frac{1}{\sqrt{2}}\left(|0\rangle + \mathrm{e}^{\left(\frac{i\pi(b' + a_{n-1}2^{n-1} + a_{n-2}2^{n-2} + \ldots + a_1 2^1 + a_0 2^0)}{2^{n-2}}\right)}\,|1\rangle\right) \quad (3.38)$$

que novamente podemos escrever como

$$\frac{1}{\sqrt{2}}\left(|0\rangle + \mathrm{e}^{\left(\underbrace{\frac{i\pi b'}{2^{n-2}}}_{\mathrm{QFT}|b'\rangle} + \underbrace{i\pi a_{n-1}\times 2}_{\text{Não faz nada}} + \underbrace{\frac{i\pi a_{n-2}}{2^0}}_{R_1\ \mathrm{ctrl}\ a_{n-2}} + \ldots + \underbrace{\frac{i\pi a_1}{2^{n-3}}}_{R_{n-2}\ \mathrm{ctrl}\ a_1} + \underbrace{\frac{i\pi a_0}{2^{n-2}}}_{R_{n-1}\ \mathrm{ctrl}\ a_0}\right)}\,|1\rangle\right), \quad (3.39)$$

e assim sucessivamente obtendo no final, após aplicar uma QFT^{-1} ao registo da soma, o estado $|a+b\rangle$. Note-se que utilizamos a palavra 'ctrl' como abreviação de *controlado por*. O circuito quântico para a adição está representado na figura 3.14, onde $|a\rangle$ e $|b\rangle$ são dois multiqubits de tamanho n. Na mesma figura está também representada a porta lógica equivalente denotada por ADD(a). Como podemos verificar ao observar este circuito, a sua estrutura é similar à da transformada de Fourier quântica o que implica que também é possível desprezar as diferenças de fase quando estas são muito pequenas, tal como vimos anteriormente. Assim a adição quântica pode ser feita utilizando também apenas $n \log_2 n$ operações (ver [28]).

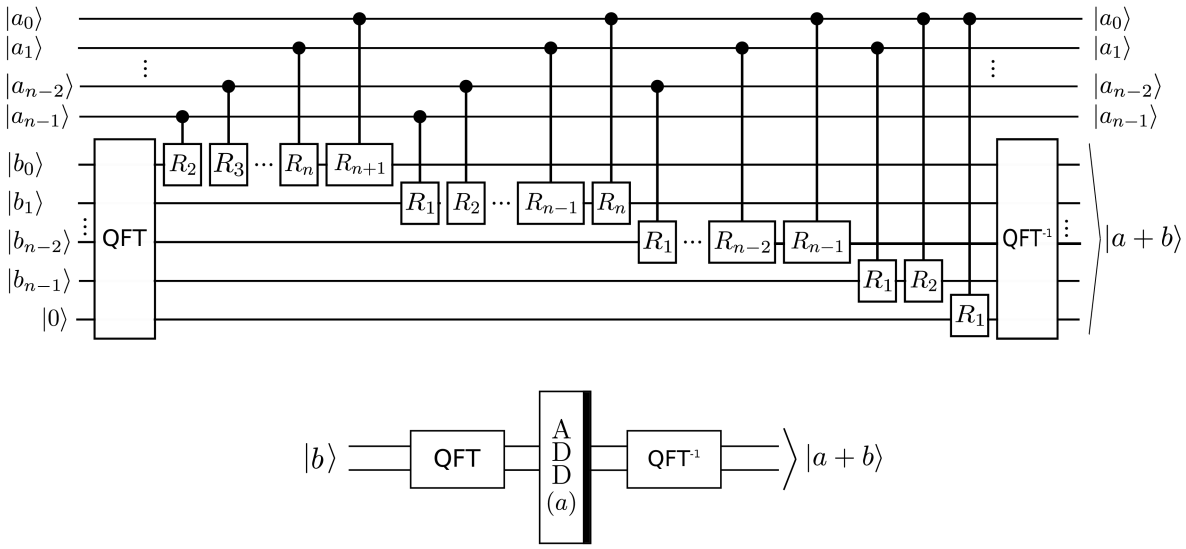


Figura 3.14: Adição quântica usando QFTs.

A subtração quântica (figura 3.15) é feita de forma idêntica à adição, com a particularidade que nesta aplicaremos as fases complexo conjugadas. O resultado será $|b-a\rangle$ se $b \geq a$ ou $\left|2^{n+1}-(a-b)\right\rangle$ se $b < a$. Se $b \geq a$, teremos que o qubit mais significativo do registo que contem a subtração será sempre $|0\rangle$ enquanto se tivermos $b < a$ o qubit mais significativo será sempre $|1\rangle$. Este resultado será importante para o próximo passo onde iremos implementar a adição módulo N, uma vez que esta última pode ser implementada numa primeira parte, como uma soma de a sempre seguida de uma subtração de N. O qubit mais significativo do resultado dir-nos-á se, antes de subtrair N, o valor contido no registo era superior ou inferior a este. Note-se que, ao longo desta dissertação, utilizaremos a porta ADD(a) com uma barra do lado direito para representar uma adição e com uma barra do lado esquerdo para representar uma subtração.

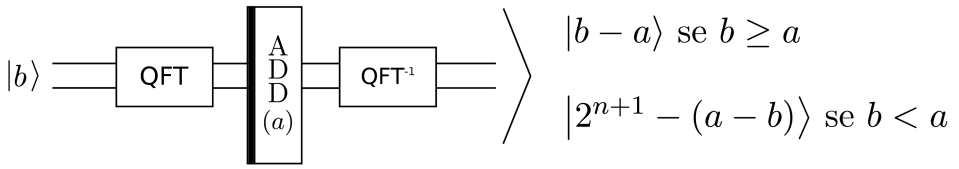


Figura 3.15: Representação da porta lógica quântica da subtração assim como os possíveis estados de saída.

Agora que já sabemos implementar uma adição normal, vamos implementar uma adição modular que denotaremos como ADD(a) mod N (figura 3.16) (ver [30, 31]). A adição módulo N implica uma subtração por N mas apenas no caso em que $a+b$ é superior a N. A forma como esse caso é avaliado não é muito simples. A subtração por N é realizada sempre (figura 3.16 a)), no final teremos que confirmar através do qubit mais significativo do resultado se antes da subtração $a+b$ era realmente maior que N. Como não podemos interagir diretamente com o sistema quântico, e consequentemente

com este qubit mais significativo, necessitamos de mais um qubit auxiliar, iniciado a $|0\rangle$, para fazer essa verificação após a aplicação de uma operação CNOT (figura 3.16 b)). Deste modo, este qubit extra será $|1\rangle$ caso se verifique que $a + b < N$. E neste último caso, teremos que voltar a adicionar N (figura 3.16 c)), voltar a colocar o qubit auxiliar no estado $|0\rangle$ revertendo todo o sistema até ao estado inicial (figura 3.16 d)) e finalmente voltar a somar a (figura 3.16 e)). Caso se verifique que $a + b > N$, então, o qubit auxiliar irá estar sempre a $|0\rangle$ e nesse caso, a parte c) do circuito, como é controlada por este, não fará nada e as partes d) e e) irão anular-se uma à outra, pois, estaremos a subtrair a para de seguida voltar a somar a, ficando assim como se, do circuito total, só se tivesse realizado a parte a). Neste circuito utilizamos duplo controlo em c_1 e c_2 pois será importante mais à frente. Assim, por agora, vamos supor que ambos se encontram no estado $|1\rangle$.

Podemos notar que neste circuito, ADD(a) mod N, não estão representados os qubits que contêm a e N para a soma e subtração. Isto acontece porque as fases necessárias para aplicar estas operações são previamente guardadas classicamente, sendo agora aplicadas diretamente sem necessitarmos de mais qubits. Este método, tem a grande vantagem de diminuir o número de qubits e portas lógicas necessárias, no entanto, caso seja necessário trocar a ou N, todas as fases terão que ser alteradas fora do computador quântico.

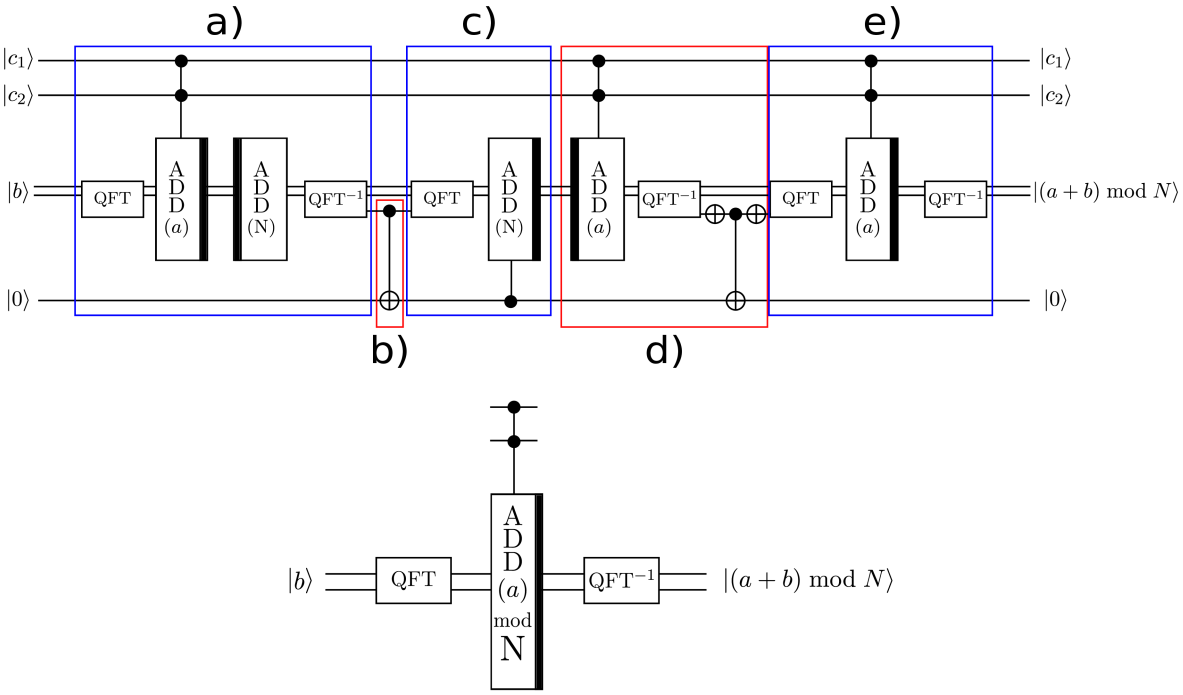


Figura 3.16: Representação do circuito e da porta lógica quântica para a adição de a mod N. a) soma a e subtrai N; b) verifica se $a + b > N$; c) soma N se $a + b < N$; d) coloca o qubit auxiliar a $|0\rangle$; e) reverte resultado após c) no registo $|b\rangle$.

Para implementarmos uma multiplicação de $(a \times b)$ mod N, assumindo que temos um a clássico, precisamos de 2 registos quânticos na seguinte ordem, $|b\rangle_n$ e $|0\rangle_n$, onde $n = \lceil \log_2 N \rceil + 1$. O procedimento é idêntico ao da adição mas neste caso o resultado da multiplicação será obtido no último registo. Para isso, vamos aplicar uma QFT a esse registo que contém $|0\rangle$ ficando com $|\phi(0)\rangle$. Escrevendo b na representação binária,

$$b = b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \ldots + b_1 2^1 + b_0 2^0, \tag{3.40}$$

a multiplicação módulo N transforma-se numa soma

$$(a \times b) \bmod N = \sum_i b_i 2^i a \bmod N. \tag{3.41}$$

Então, primeiro efetuamos ADD(a) mod N controlado por b_0, ficando

$$\left|\phi(0 + (ab_0 2^0) \bmod N)\right\rangle, \tag{3.42}$$

depois ADD($2a$) mod N, mas desta vez, controlado por b_1, resultando em

$$\left|\phi(0 + (ab_0 2^0 + ab_1 2^1) \bmod N)\right\rangle. \tag{3.43}$$

Procedendo sempre desta forma até termos adicionado um *adder* controlado em cada qubit de $|b\rangle$, ficaremos com

$$\begin{aligned}\left|\phi(0 + (ab_0 2^0 + ab_1 2^1 + \ldots + ab_{n-2} 2^{n-2} + ab_{n-1} 2^{n-1}) \bmod N)\right\rangle &= |\phi(0 + (a \times b) \bmod N)\rangle \\ &= |\phi((a \times b) \bmod N)\rangle. \end{aligned} \tag{3.44}$$

Por último, para obtermos a multiplicação $(a \times b)$ mod N, só teremos que aplicar uma QFT^{-1} neste último registo ficando com

$$\mathrm{QFT}^{-1} |\phi((a \times b) \bmod N)\rangle = (a \times b) \bmod N. \tag{3.45}$$

3.5 Algoritmo de Shor

Em 1993, Shor mostrou que um computador quântico conseguiria fatorizar números inteiros compostos em tempo polinomial. Ele usou o facto do problema da fatorização poder ser resolvido encontrando o período de certas funções (ver [20]) o qual se pode realizar quanticamente num processo cujo tempo de computação cresce polinomialmente com o número a fatorizar, usando o paralelismo quântico. A este algoritmo deu-se o nome de algoritmo de Shor.

Para o caso particular de encontrar dois fatores primos, p e q, de N tal que $N = pq$, basta encontrar o período da seguinte exponenciação modular

$$f(x) = a^x \bmod N \tag{3.46}$$

onde a é um inteiro qualquer menor e co-primo com N. Encontrar este período é o equivalente a descobrir qual a $\mathrm{ord}_N(a)$. Assim, o processo de fatorização quântico consistirá em usar o algoritmo de Shor para encontrar a $\mathrm{ord}_n(a)$ sendo o restante processo de fatorização realizado de forma clássica.

Nas próximas subsecções iremos descrever, acompanhando com exemplos, o algoritmo de Shor, de um modo clássico e de um modo quântico. Como a implementação quântica é a mais relevante pois é esta que o torna numa ameaça à maioria dos sistemas clássicos, esta será descrita mais detalhadamente. O circuito quântico para este algoritmo também será apresentado.

3.5.1 Algoritmo - método clássico

A fatorização de um número composto $N = pq$, onde $p \neq q$, utilizando o algoritmo de Shor consiste na realização dos seguintes passos:

1. Escolher um inteiro a tal que $\mathrm{mdc}(a, N) = 1$ e $a < N$;

2. Calcular $f(x) = a^x \bmod N$ com $x = 1, 2, \ldots, N-1$ e descobrir o primeiro x para o qual $f(x) = 1$. Este resultado é o período de $f(x)$, não é mais que a $\mathrm{ord}_N(a)$, aqui designado por r;

3. Caso r seja ímpar ou $a^{\frac{r}{2}} = -1 \bmod N$, recomeçar o algoritmo escolhendo outro a. Está demonstrado em [19, 32] que em pelo menos metade dos casos isto irá acontecer.

4. Caso contrário, fazer:

$$P = a^{\frac{r}{2}} + 1 \tag{3.47}$$

$$Q = a^{\frac{r}{2}} - 1. \tag{3.48}$$

5. Por último calcular:

$$p = \text{mdc}(P, N) \tag{3.49}$$
$$q = \text{mdc}(Q, N). \tag{3.50}$$

Se p ou q for um fator não trivial de N então a fatorização de N foi bem sucedida. Caso isto não se verifique, repetir todo o processo para outro a.

O exemplo seguinte é realizado inteiramente de forma clássica. No entanto, classicamente este algoritmo não pode ser resolvido em tempo polinomial devido à exponenciação modular do passo 2, onde o número de vezes que esta terá que ser realizada aumenta exponencialmente com $\log_2 N$. Como existe um algoritmo que resolve esta exponenciação em tempo polinomial num computador quântico, esta parte será realizada quanticamente.

Exemplo: Escolhemos $N = pq = 21$ e $a = 11$.

Calculamos o período fazendo:

$$\begin{gathered} 11^1 \bmod N = 11 \\ 11^2 \bmod N = 16 \\ 11^3 \bmod N = 8 \\ 11^4 \bmod N = 4 \\ 11^5 \bmod N = 2 \\ 11^6 \bmod N = 1. \end{gathered}$$

Logo o período é $r = 6$.

Calculamos P e Q:

$$\begin{aligned} P &= 11^{\frac{6}{2}} + 1 = 1332 \\ Q &= 11^{\frac{6}{2}} - 1 = 1330 \end{aligned}$$

Finalmente obtemos os fatores de N, fazendo:

$$\begin{aligned} p &= \text{mdc}(P, N) = 3 \\ q &= \text{mdc}(Q, N) = 7. \end{aligned}$$

3.5.2 Algoritmo - parte quântica

Os circuitos quânticos que implementam o algoritmo de Shor podem ser construídos de forma a minimizar o número de qubits necessários, tempo de execução, facilidade de construção ou uma combinação destes fatores. De um modo geral, quanto menor o número de qubits maior o número de portas lógicas que teremos que implementar e consequentemente maior o tempo de execução do algoritmo [31]. Na nossa abordagem usaremos um método que, como já percebemos pela maneira como implementamos a adição e a multiplicação, tentará minimizar o número de qubits. Assim, para um caso geral, a implementação do algoritmo de Shor segundo a nossa abordagem requer $L_1 + 2L_2 + 3$ qubits, onde L_1 corresponde ao número de qubits necessários para representar x tal que $N^2 < 2^{L_1} < 2N^2$ e L_2 corresponde ao número de qubits necessários para representar N (ver [2]). Na figura 3.17 temos uma representação simplificada do circuito da parte quântica correspondente ao algoritmo de Shor que iremos implementar mais à frente. Nesta representação temos dois registos quânticos iniciados a $|0\rangle$. No primeiro registo, após a aplicação das Hadamards, iremos ter uma sobreposição de todos os valores possíveis de x enquanto que no segundo registo, iremos ter uma função $f(x)$, que representa a exponenciação modular $a^x \bmod N$, que se consegue por aplicação de portas com controlo no primeiro registo. Finalmente será aplicada uma QFT^{-1} no primeiro registo e feita a medição que nos dará o período r de $f(x)$, ou um outro valor da qual poderemos, em alguns casos, obter de forma indireta este r. Todos este procedimento será pormenorizado mais à frente.

Consideremos a exponenciação modular em 3.46. Vamos criar 2 registos quanticos: $|0\rangle_{L_1}$ para o x e $|0\rangle_{L_2+1}$ que servirá para calcular e armazenar o resultado de $f(x)$. Mais à frente, quando explicarmos a exponenciação modular quântica, iremos precisar de mais dois registos. Um $|b\rangle_{L_2+1}$, de tamanho $L_2 + 1$, que irá ser utilizado para os cálculos de $f(x)$ e um outro registo $|0\rangle_1$, de tamanho 1, que

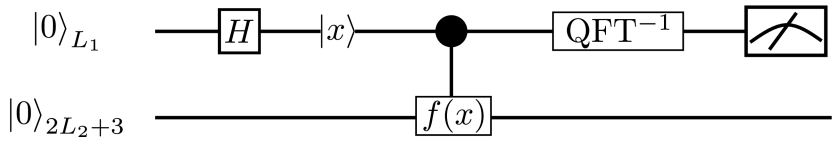


Figura 3.17: Circuito quântico para encontrar a ordem $\mathrm{ord}_N(a)$, $f(x)$ implementa $a^x \bmod N$.

servirá para controlos. No entanto, para a explicação da parte quântica do algoritmo, omitiremos esses últimos que só serão utilizados para cálculos. Consideremos então os 2 registos

$$|0\rangle_{L_1} |0\rangle_{L_2+1} . \tag{3.51}$$

Vamos aplicar Hadamards a todo o primeiro registo, ficando lá com uma sobreposição equiprovável de 2^{L_1} estados, ou seja, teremos os registos como

$$\frac{1}{\sqrt{2^{L_1}}} \sum_{x=0}^{2^{L_1}-1} |x\rangle_{L_1} |0\rangle_{L_2+1} . \tag{3.52}$$

De seguida vamos fazer a exponenciação modular, onde o expoente será o primeiro registo, e guardar o seu valor no segundo registo. Assim teremos

$$\frac{1}{\sqrt{2^{L_1}}} \sum_{x=0}^{2^{L_1}-1} |x\rangle_{L_1} |f(x)\rangle_{L_2+1} . \tag{3.53}$$

A medição seguinte pode ser dispensada, no entanto, iremos fazê-la na mesma porque é útil para a explicação. Vamos medir o segundo registo e obter um valor f_α. Esta medição fará com que o primeiro registo colapse para um estado de sobreposição de todos os valores de x para o qual $f(x) = f_\alpha$. Como $f(x)$ é periódica em x com ordem r, então após esta medição teremos

$$\frac{\sqrt{r}}{\sqrt{2^{L_1}}} \sum_{n=0}^{\frac{2^{L_1}}{r}-1} |x_\alpha + nr\rangle_{L_1} |f_\alpha\rangle_{L_2+1} , \tag{3.54}$$

onde x_α é o valor mais baixo de x que é compatível com f_α. Como podemos observar, ao medirmos o segundo registo ele colapsou para um dos possíveis estados, logo, como nós não temos cópias deste registo antes da medição pois, como já visto anteriormente, é impossível clonar um estado quântico, a partir dele não podemos obter qualquer informação acerca de r. No entanto, podemos aplicar uma QFT ao primeiro registo obtendo

$$\frac{\sqrt{r}}{2^{L_1}} \sum_{n=0}^{\frac{2^{L_1}}{r}-1} \sum_{k=0}^{2^{L_1}-1} \mathrm{e}^{\frac{2\pi i k(x_\alpha + nr)}{2^{L_1}}} |k\rangle_{L_1} |f_\alpha\rangle_{L_2} . \tag{3.55}$$

Por fim, ao medirmos agora o primeiro registo, a probabilidade de lermos k é dada por

$$\begin{aligned} P(k) &= \left| \frac{\sqrt{r}}{2^{L_1}} \sum_{n=0}^{\frac{2^{L_1}}{r}-1} \mathrm{e}^{\frac{2\pi i k(x_\alpha + nr)}{2^{L_1}}} \right|^2 \\ &= \left| \frac{\sqrt{r}}{2^{L_1}} \mathrm{e}^{\frac{2\pi i k x_\alpha}{2^{L_1}}} \sum_{n=0}^{\frac{2^{L_1}}{r}-1} \mathrm{e}^{\frac{2\pi i k n r}{2^{L_1}}} \right|^2 . \end{aligned} \tag{3.56}$$

A parte $e^{\frac{2\pi i k x_\alpha}{2^{L_1}}}$ de 3.56 é apenas uma fase cujo módulo é 1, logo, para esta medição temos dois casos possíveis:

- **Caso 1 -** $\frac{2^{L_1}}{r}$ é inteiro, ou seja, r é uma potência de 2.

  Se, $\frac{kr}{2^{L_1}}$ for inteiro, isto é, k múltiplo de $\frac{2^{L_1}}{r}$, na forma

  $$k = c\frac{2^{L_1}}{r}, \tag{3.57}$$

  com c inteiro tal que $0 \leq c < r$. Neste caso $e^{\frac{2\pi i n k r}{2^{L_1}}} = 1$ para qualquer que seja o n, logo

  $$P(k) = \left| \frac{\sqrt{r}}{2^{L_1}} \times \frac{2^{L_1}}{r} \right|^2 = \frac{1}{r}. \tag{3.58}$$

  Há r casos em que k é múltiplo de $\frac{2^{L_1}}{r}$, logo, há r k's que podem ser lidos com probabilidade $\frac{1}{r}$ cada. Como a soma de todas estas r probabilidades é 1, então todos os outros k's tem probabilidade nula de serem medidos. Deste modo, para este caso em que $\frac{2^{L_1}}{r}$ é um inteiro, obteremos sempre um valor de k de acordo com 3.57.

- **Caso 2 -** Se $\frac{2^{L_1}}{r}$ não é um inteiro, pode provar-se que as medidas mais prováveis são

  $$k \simeq c\frac{2^{L_1}}{r}, \tag{3.59}$$

  com $0 \leq c < r$, onde $\left| k - c\frac{2^{L_1}}{r} \right| < \frac{1}{2}$ (ver [2]). Aqui termina a parte quântica do algoritmo. Os seguintes passos serão clássicos. Vamos reescrever a equação 3.59 como

  $$\frac{k}{2^{L_1}} \simeq \frac{c}{r}. \tag{3.60}$$

  Como apenas sabemos o valor de k e L_1, teremos que estimar valores inteiros para c e r, onde $r < N$, tais que $\frac{c}{r}$ seja muito próximo de $\frac{k}{2^{L_1}}$. Para isso, podemos escrever $\frac{k}{2^{L_1}}$ na sua representação em frações contínuas, que iremos explicar detalhadamente em seguida, obtendo assim uma representação do tipo $\frac{k}{2^{L_1}} = [z_0; z_1, \ldots, z_j]$. Assim, a partir desta representação, podemos ir escrevendo os seus convergentes $\frac{p_j}{q_j}$ com p_j e q_j dados por

  $$\frac{p_0}{q_0} = \frac{z_0}{1} \tag{3.61}$$

  $$\frac{p_1}{q_1} = \frac{z_0 z_1 + 1}{z_1}, \tag{3.62}$$

  e para $j > 1$

  $$\frac{p_j}{q_j} = \frac{p_{j-1} z_j + p_{j-2}}{q_{j-1} z_j + q_{j-2}}. \tag{3.63}$$

  Após termos estes convergentes, o nosso período r será da forma $q_j t$ onde t é um inteiro positivo. Finalmente, só teremos que calcular sistematicamente $a^{q_j t} \bmod N$, para todos os q_j e t tal que $q_j t < N$. Quando obtivermos $a^{q_j t} \bmod N = 1$ teremos encontrado o nosso período $r = q_j t$.

Método da expansão em frações contínuas

Como vimos na secção anterior nem sempre o algoritmo de Shor, quando implementado quanticamente, nos dá diretamente o período, mas sim um valor que em alguns casos pode ser trabalhado usando o método da expansão em frações finitas de modo a que, a partir da representação dessa expansão, se consiga chegar ao período. Assim, explicaremos agora este método, que consiste

numa forma de expressar um número real e racional R como

$$R = z_0 + \cfrac{1}{z_1 + \cfrac{1}{\ddots + \cfrac{1}{z_{j-1}}}} \tag{3.64}$$

onde $z_i \in \mathbb{N}$. Vamos assumir que R pode ser expresso como uma fração contínua finita. Então, para o expressarmos nessa forma, teremos que:

- escrever R é racional, logo, como uma soma de uma parte inteira z_0 com uma parte fracionária f_0 ficando com $R = z_0 + f_0$;
- pegar na parte f_0 e expressar R como $R = r + \frac{1}{\frac{1}{f_0}}$;
- escrever $\frac{1}{f_0}$ como uma soma da sua parte inteira z_1 com a sua fracionária f_1;
- continuar este procedimento até que a parte fracionária se torne zero.

Na representação das frações contínuas escrevemos R como $[z_0; z_1, \ldots, z_j]$. Para melhor perceber este algoritmo vamos ilustrá-lo com o seguinte exemplo: vamos considerar $R = \frac{43}{32}$. Então

$$\begin{aligned}
\frac{43}{32} &= 1 + \frac{11}{32} \\
&= 1 + \cfrac{1}{\cfrac{32}{11}} \\
&= 1 + \cfrac{1}{2 + \cfrac{10}{11}} \\
&= 1 + \cfrac{1}{2 + \cfrac{1}{\cfrac{11}{10}}} \\
&= 1 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{10}}}.
\end{aligned}$$

Logo, $R = \frac{43}{30} = [1; 2, 1, 10]$ e os seus convergentes, ordenados, são $\left\{1, \frac{3}{2}, \frac{4}{3}, \frac{43}{32}\right\}$.

Exponenciação modular quântica

Na subsecção 3.4.2 mostrámos como realizar operações mais básicas como a adição e a multiplicação modular. Aqui descreveremos a exponenciação modular. De entre os vários métodos possíveis para a sua implementação, iremos utilizar um baseado na adição Draper descrito em cima pois é o que requer um menor número de qubits. Seja L_1 o número de bits necessários para representar x. Então, de acordo com 2.14 podemos escrever

$$\begin{aligned}
f(x) &= a^x \bmod N \\
&= \left[\prod_{i=0}^{L_1-1} (a^{2^i x_i} \bmod N)\right] \bmod N,
\end{aligned} \tag{3.65}$$

onde

$$\sum_{i=0}^{L_1-1} (2^i x_i) = x \tag{3.66}$$

é a decomposição binária de x, ou seja, x_i é o bit de ordem i de x. A exponenciação modular ficou assim escrita como o produto de L_1 termos. No entanto, apenas teremos que multiplicar $a^{2^i x_i} \bmod N$ se

$x_i = 1$. Caso contrário nada acontecerá. Vamos agora definir exponenciações modulares incompletas ou parciais da forma

$$f_j(x) = \left[\prod_{i=0}^{j-1} (a^{2^i x_i} \bmod N)\right] \bmod N. \tag{3.67}$$

Vamos também definir

$$g_k(x) = a^{2^k x_k} \bmod N \tag{3.68}$$

como o termo de ordem k do produtório 3.65. Consideremos que já temos $f_j(x)$ e que pretendemos multiplicar o termo seguinte $g_{j+1}(x)$ caso $x_{j+1} = 1$. No nosso circuito, teremos então $|f_j(x), 0\rangle$ denotando dois registos quânticos com o mesmo tamanho, um contendo $f_j(x)$ e outro o 0. Se $x_{j+1} = 1$, o próximo passo do algoritmo para a exponenciação modular pode ser dividido nos seguintes passos:

1. adicionar, ao segundo registo, $g_{j+1}(x)$ modularmente multiplicado pelo primeiro registo, ou seja,

$$\begin{aligned} |f_j(x), 0\rangle &\longmapsto |f_j(x), 0 + (g_{j+1}(x) f_j(x)) \bmod N\rangle \\ &= |f_j(x), f_{j+1}(x)\rangle\,; \end{aligned} \tag{3.69}$$

2. trocar os dois registos (SWAP), ficando

$$|f_{j+1}(x), f_j(x)\rangle \tag{3.70}$$

3. colocar o segundo registo novamente a 0 subtraindo ao segundo registo, $(g_{j+1}(x))^{-1}$ multiplicado modularmente pelo primeiro registo, ou seja,

$$\begin{aligned} |f_{j+1}(x), f_j(x)\rangle &\longmapsto \left|f_{j+1}(x), f_j(x) - ((g_{j+1}(x))^{-1} f_{j+1}(x)) \bmod N\right\rangle \\ &= |f_{j+1}(x), 0\rangle\,. \end{aligned} \tag{3.71}$$

Se $x_{j+1} = 0$, não acontecerá nada e ficaremos com os estados anteriores, neste caso os iniciais, $|f_j(x), 0\rangle = |f_{j+1}(x), 0\rangle$.

As multiplicações, do primeiro e terceiro passo do parágrafo anterior, iremos dividi-las em adições e subtrações controladas da forma

$$f_j(x) g_{j+1}(x) = \sum_{i=0}^{L_1-1} g_{j+1} 2^i (f_j)_i \bmod N, \tag{3.72}$$

$$(g_{j+1}(x))^{-1} f_{j+1}(x) = \sum_{i=0}^{L_1-1} (g_{j+1})^{-1} 2^i (f_{j+1})_i \bmod N, \tag{3.73}$$

onde $(f_j)_i$ e $(f_{j+1})_i$ correspondem ao bit de ordem i de (f_j) e (f_{j+1}) respetivamente.

Por fim, para percebermos melhor este algoritmo, vamos ilustrá-lo com um exemplo para encontrar a ordem de $f(x) = a^x \bmod N$ usando a exponenciação modular descrita em cima. Como já visto anteriormente, cada operação requer uma grande quantidade de portas lógicas e portanto, vamos escolher um número $N = pq$ pequeno e um a, que como vimos tem de ser coprimo com N, que resulte num período pequeno, de modo a facilitar a implementação. Assim, escolhemos $N = 15$ e $a = 4$, pois uma vez que $\mathrm{ord}_{15}(4) = 2$ basta que x tenha 2 qubits. Como são necessários 4 bits para representar N, necessitaremos de um registo com 5 qubits para aplicar as subtrações e adições por N da exponenciação modular. Como já visto anteriormente, o registo que contém $f(x)$ terá que ter o mesmo tamanho que o que contém a exponenciação, logo 5 qubits. Por último, precisaremos também de um qubit extra para auxiliar nas adições modulares. Deste modo, teremos que ter um registo quântico com um total de 13 qubits, todos inicializados a $|0\rangle$, dos quais 2 representarão x, 5 representarão $f(x)$, 5 representarão o registo onde será feita a exponenciação modular e que denotaremos por b, e 1 qubit extra para auxiliar nas adições modulares. Assim, o circuito quântico do algoritmo de Shor para estes parâmetros com as respetivas fases já colocadas diretamente encontra-se representado na figura 3.18. Neste circuito, tal como descrito anteriormente, aplicamos Hadamards em todo o primeiro registo de modo a termos lá uma sobreposição de todos os valores de x possíveis. O primeiro qubit do registo $|f(x)\rangle$, foi colocado a

$|1\rangle$ pois, este é o elemento neutro da multiplicação e, se repararmos, o primeiro valor da exponenciação modular teria que ser sempre 1. Além disso, como todas as operações que iremos realizar tem controlo nesse registo, se ele fosse todo $|0\rangle$, nenhuma delas seria realizada. Após isto, entre a primeira QFT e respetiva inversa, é realizada a primeira série de somas, controladas pelo registo de $|f(x)\rangle$ e pelo primeiro qubit de $|x\rangle$. Depois, é feito um SWAP entre os valores dos registos $|b\rangle$ e $|f(x)\rangle$, controlado pelo primeiro qubit de $|x\rangle$. No final teremos a primeira exponenciação modular. De seguida, teremos que efetuar a segunda e última série de adições, com controlo no registo $|f(x)\rangle$ e segundo qubit de $|x\rangle$. No entanto, antes disso precisamos nos certificar que o registo de $|b\rangle$ está a zeros. Para isso iremos fazer as subtrações controladas que se observam logo após a segunda QFT. As adições controladas serão feitas após estas subtrações. Depois, voltamos a fazer o SWAP entre os mesmos registos mas desta vez com controlo no segundo qubit de $|x\rangle$ e voltamos a colocar o registo $|b\rangle$ a zeros, obtendo a exponenciação modular completa. Por último, é aplicada uma QFT^{-1} em $|x\rangle$ e efetuada a medição. Após esta implementação, o resultado obtido na medição foi em aproximadamente metade das vezes $|00\rangle$ e na outra metade $|10\rangle$ que em decimal correspondem ao resultado 0 e 2, respetivamente. Este resultado faz todo o sentido, pois, como já sabemos de antemão que o período é 2, r é uma potência de 2, logo estamos no caso 1 visto anteriormente, então o resultado da medição k será sempre da forma em 3.57 onde, neste caso, sabemos que c só poderá tomar os valores $\{0, 1\}$ e sabemos o valor de $2^{L_1} = 2^2 = 4$. Ao substituirmos estes valores na equação 3.57 teremos um valor de k para cada valor de c, ou seja, para $c = 0$ teremos $k = 0$, para $c = 1$ teremos $k = 1 \times \frac{4}{2} = 2$, o que corresponde aos dois valores que obtivemos. A restante parte da fatorização é feita de forma clássica. Assim, ao substituir r e os parâmetros utilizados em 3.47 e 3.48, obteremos $P = 5$ e $Q = 3$, que posteriormente ao substituir em 3.49 e 3.50 nos devolverá os dois fatores de N, $p = 5$ e $q = 3$.

Na implementação computacional, adaptámos um simulador quântico da biblioteca *qiskit* fornecida pelo IBM, e programámos em python. Como o programa ainda demorava algum tempo até finalizar a execução, realizámos apenas 25 iterações.

3.6 Conclusões

O algoritmo de Shor é um algoritmo de fatorização que é polinomial quando implementado num computador quântico. No entanto, como já discutimos na introdução e na secção 3.3, ainda há um longo caminho a percorrer até que possamos ter computadores quânticos capazes de usar este algoritmo sistematicamente e para números realmente grandes.

A construção deste capítulo exigiu, além de um estudo sobre os fundamentos da física e computação quântica, a aprendizagem de um novo modo de programação. Toda a programação, apesar de ser realizada em python, é feita com a manipulação de portas lógicas, como se fosse num computador quântico real, sendo assim de muito baixo nível. Qualquer operação necessitava de ser implementada de raiz. Daí termos começado com a implementação das transformadas de Fourier quânticas, depois as adições, subtrações, adições modulares, e assim sucessivamente até chegarmos ao objetivo final que era implementar o algoritmo de Shor. À medida que cada operação ia sendo implementada, esta tinha de ser constantemente testada, a cada passo, de forma clássica, para termos a certeza que em cada passo o resultado estava correto e que podíamos avançar para o seguinte. Inicialmente, tentamos programar no computador quântico da IBM. No entanto, só tínhamos acesso a 5 qubits e como nem todas as ligações entre eles era possível, o máximo que conseguimos foi implementar uma QFT de terceira ordem. Apesar disso, com base no simulador deles e adaptando a biblioteca *qiskit* que eles disponibilizavam, foi possível criar o nosso próprio simulador em python, implementar todos os circuitos presentes neste capítulo (ver [12]) e consequentemente cumprir o nosso objetivo, fatorizar um número usando o algoritmo de Shor quântico.

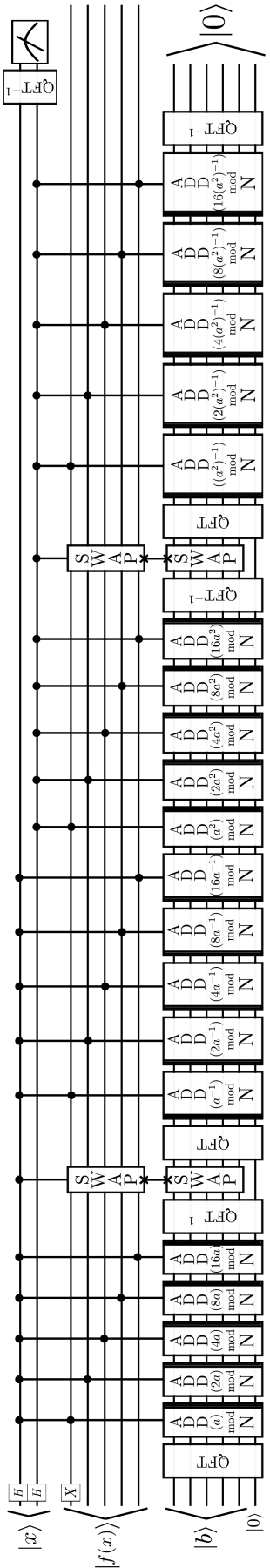


Figura 3.18: Circuito quântico do algoritmo de Shor, simplificado para $N = 15$ e $a = 4$.

Capítulo 4

Criptografia Pós-Quântica

Beauty is the first test: there is no permanent place in the world for ugly mathematics.

G. H. Hardy

Como já foi discutido na introdução e no capítulo anterior, com a evolução da computação quântica, nomeadamente dos computadores quânticos, surge a ameaça aos sistemas de segurança convencionais. Através de um computador quântico o algoritmo de Shor consegue quebrar, em tempo polinomial, qualquer criptossistema clássico cuja segurança se baseia no problema da fatorização ou no problema do logaritmo discreto (ver [33]), como é caso do RSA, ElGamal, DSA ou ECDSA. Assim, como uma resposta a esta ameaça, com o intuito de estudar e criar alternativas a estes criptossistemas clássicos, surge a criptografia pós-quântica.

Entre os criptossistemas pós-quânticos mais promissores, encontram-se os baseados em códigos corretores de erros. Em particular vamo-nos focar no criptossistema de McEliece para o qual, até ao momento, não existe nenhum algoritmo quântico, e não quântico, capaz de o quebrar. Neste capítulo iremos descrever este criptossistema assim como uma nova variante do mesmo baseada em códigos convolucionais. Como tal, começaremos por introduzir alguns conceitos sobre teoria dos códigos e sobre possíveis ataques a estes criptossistemas.

4.1 Criptografia baseada em Códigos

Este tipo de criptografia tem como principal característica a utilização de códigos que permitem a correção de um número limitado de erros. A seguir, explicamos os conceitos básicos dos códigos corretores de erros e damos exemplos de alguns desses códigos.

4.1.1 Teoria dos Códigos

Durante uma transmissão digital de dados o próprio canal de transmissão pode tornar-se ruidoso devido a, por exemplo, interferências eletromagnéticas, que podem introduzir, com uma certa probabilidade, alguns erros no sinal. Deste modo, torna-se importante ter um mecanismo que permita a localização e correção de erros de maneira a que, se o sinal não se encontrar demasiado corrompido, se consiga recuperar os dados originais. A solução para este problema teve início no final da década de 40 quando Richard Hamming desenvolveu um código capaz de detetar até dois erros e corrigir um, se este for único. Pouco tempo depois, Claude Shannon, juntamente com o trabalho já desenvolvido por Hamming, apresentou duas novas áreas de pesquisa: a Teoria dos Códigos e a Teoria da Informação. Desde então milhares de artigos sobre estes assuntos foram publicados.

Para que os erros possam ser detetados e corrigidos, os dados devem ser codificados usando estes códigos corretores de erros. Estes codificam a informação inicial, adicionando informação redundante, de modo a que, após esta ser afetada pelo ruído durante a sua transmissão, seja possível através de um algoritmo de descodificação recuperar a informação original. Notemos que, para que um processo de descodificação seja sempre bem sucedido, a codificação tem que ser uma função injetiva onde a

cada símbolo, ou conjunto de símbolos codificados como um bloco, corresponde uma única respetiva palavra de código.

Corpos finitos

Seja a *característica* de um corpo $\mathbb{F}$ o menor inteiro positivo r tal que $r \cdot a = 0$ para qualquer $a \in \mathbb{F}$. Seja p um primo e $m \in \{1, 2, \ldots\}$. Então, existe um corpo finito denotado por $\mathbb{F}_q$, com $q = p^m$ elementos, onde q é designado por *ordem* e p a sua *característica*. Estes corpos também são frequentemente chamados de corpos de Galois, sendo denotados por $GF(q)$. Para $m \geq 2$ diz-se que $GF(p^m)$ é uma *extensão* de $GF(p)$.

Assim, um corpo finito não é mais que um conjunto finito de elementos $GF = \{A, B, C, \ldots\}$ munido por leis da composição interna (adição e multiplicação) que devem satisfazer os seguintes requisitos:

1. O resultado de operações entre elementos do corpo é elemento do corpo;
2. Um dos elementos, I, é o elemento identidade tal que, para qualquer elemento A do corpo, $AI = IA = A$;
3. Lei da associatividade: $A(BC) = (AB)C$;
4. Lei da distributividade: $(A + B)C = AC + BC$;
5. Ter elemento nulo, 0, tal que: para qualquer elemento A, $A0 = 0A = 0$;
6. Possui elemento simétrico para a adição, isto é, para qualquer elemento A pertencente ao corpo, existe um elemento B também pertencente ao corpo tal que, $A + B = 0$;
7. Todo o elemento $A \neq 0$ pertencente ao corpo possui inverso tal que $AA^{-1} = I$. Este A^{-1} é chamado de inverso multiplicativo de A e pode ser calculado através do algoritmo de Euclides.

Um exemplo para a soma e multiplicação de um corpo finito $GF(4) = GF(2^2) = \{0, 1, \alpha, \beta\}$ pode ser observado na tabela tabela 4.1)

$+$	0	1	α	β
0	0	1	α	β
1	1	0	β	α
α	α	β	0	1
β	β	α	1	0

$\cdot$	0	1	α	β
0	0	0	0	0
1	0	1	α	β
α	0	α	β	1
β	0	β	1	α

Tabela 4.1: Soma e multiplicação entre elementos de um corpo finito $GF(2^2)$.

Vamos agora supor que temos um polinómio irredutível (que não pode ser fatorizado) e primitivo (que o máximo divisor comum entre os seus coeficientes é 1) $f(x)$ de grau m. Então pode ser construído um corpo $GF(p)[x]$ sobre esse polinómio onde cada elemento $GF(p)[x]/(f(x))$ é um polinómio de grau máximo $m - 1$.

Como exemplo vamos escrever $GF(8) = GF(2^3)$ e considerar $f(x)$ um polinómio irredutível e primitivo sobre este corpo, por exemplo, $f(x) = x^3 + x + 1$. Logo à partida sabemos que o corpo tem 8 elementos módulo 2 de ordem máxima $3 - 1 = 2$, logo, cada elemento na forma polinomial será da forma $ax^2 + bx + c1$ onde a, b e c são dígitos binários podendo cada elemento do corpo ser representado pelos 3 bits abc. Assim, vamos identificar $GF(2^3)$ como $GF(2)[x]/(f(x))$. Para gerar os elementos de $GF(2^3)$ a partir de $f(x)$, começamos com os elementos $\{0, 1\}$ e seguimos multiplicando o último elemento por x fazendo de seguida módulo $f(x)$. Assim, como cada elemento de $GF(2^3)$, além de módulo 2, é também módulo $f(x)$, podemos estabelecer que $f(x) = 0$ ficando assim $x^3 + x + 1 = 0$, ou seja, $x^3 = x + 1$. Deste modo é possível reescrever o corpo em função das potências de x como $GF(2)[x]/(f(x)) = \{0, 1, x, x^2, x^3, x^4, x^5, x^6\}$ e estabelecer a correspondência de cada potência com o respetivo polinómio e forma binária. Este exemplo encontra-se representado na tabela 4.2.

Um outro conceito importante é o de *anel*. De forma simples um *anel* é uma estrutura algébrica munida pelas operações de adição e multiplicação, onde os seus elementos devem satisfazer as condições enunciadas anteriormente para os corpos finitos, com exceção da número 7.

potência	polinomial	vetorial
0	0	000
$x^0 = x^7 = 1$	1	001
x^1	x	010
x^2	x^2	100
x^3	$x+1$	011
x^4	x^2+x	110
x^5	x^2+x+1	111
x^6	x^2+1	101

Tabela 4.2: Formas de representação de um corpo $GF(8)$.

Adição e multiplicação sobre $GF(p^m)$

Devido às propriedades dos corpos finitos já estudadas podemos questionar: como serão feitas as adições e multiplicações entre elementos do mesmo corpo, dado que este possui um número finito de elementos? A resposta é simples. Caso a característica do corpo seja 2, a adição pode ser facilmente obtida através da representação vetorial usando a operação XOR, no entanto, para um caso geral, a adição é feita através da representação polinomial dos elementos, fazendo módulo p. A multiplicação pode ser feita usando a forma polinomial e no final, caso o grau do polinómio obtido seja igual ou superior ao do polinómio primitivo, deverá fazer-se a sua redução fazendo o módulo com o mesmo polinómio primitivo gerador do corpo. A divisão é idêntica à multiplicação se tivermos em conta que $A/B = AB^{-1}$, assim, uma divisão é o mesmo que uma multiplicação pelo inverso multiplicativo. Cada operação pode ser realizada em qualquer uma das representações, no entanto, o método pode tornar-se complexo, e por isso, iremos apenas explicar cada operação para a representação onde cada operação é mais simples. Como para o exemplo seguinte o corpo é de característica 2, iremos também realizar a soma através da representação vetorial.

Para exemplificar, consideremos o caso representado na tabela 4.2.

- Adição: $x^4 + x^6$

$$\begin{aligned} x^4 + x^6 &= x^2 + x + x^2 + 1 \\ &= 2x^2 + x + 1 \\ &= x + 1 \end{aligned}$$

na forma binária:

$$\begin{array}{rr} & 110 \\ + & 101 \\ \hline & 011 \end{array} \quad = x+1$$

- Multiplicação: $x^4 \times x^6$

$$\begin{aligned} x^4 \times x^6 &= (x^2 + x) \times (x^2 + 1) \\ &= x^4 + x^2 + x^3 + x \\ &= x^2 + x + x^2 + x + 1 + x \\ &= x + 1 \end{aligned}$$

ou

$$\begin{aligned} x^4 \times x^6 &= x^{10} \\ &= x^7 \times x^3 \\ &= 1 \times x^3 \\ &= x + 1 \end{aligned}$$

Códigos lineares

Seja $\mathbb{F}_q$ um corpo finito de ordem q. Então define-se como *código linear* de comprimento n, a um subespaço vetorial $\mathcal{C}$ de $\mathbb{F}_q^n$. Os vetores em $\mathcal{C}$ são denominados de *palavras de código*. Como $\mathcal{C}$ é um subespaço de $\mathbb{F}_q^n$, então existe uma base, com $k \leq n$ elementos, pertencente a $\mathbb{F}_q^n$, onde k representa a dimensão desse subespaço. Deste modo, esse código linear é denotado por $\mathcal{C} - [n, k]$ e pode ser representado matricialmente de duas formas. Uma delas é através da matriz geradora do código e outra é através da matriz paridade. A matriz geradora, que denotaremos por G, é do tipo $k \times n$ tal que cada palavra de código, c, pode ser expressa como uma combinação linear das suas linhas, isto é, as suas linhas formam uma base de $\mathcal{C}$. Deste modo,

$$\mathcal{C} = \{c = uG \mid \forall u \in \mathbb{F}^k\} = \mathrm{Im}_{\mathbb{F}_q} G \subset \mathbb{F}_q^n, \tag{4.1}$$

onde u é o vetor informação. Logo, esta é a matriz utilizada na codificação, ou seja, é a que transforma uma mensagem u de tamanho k numa palavra de código c de tamanho n. Assim, dizemos que codificamos u em c. Já a matriz paridade, que denotaremos por H, é do tipo $(n-k) \times n$, de característica $(n-k)$, tal que $Hc^T = 0 \ \forall \ c \in \mathcal{C}$, ou seja, dado que cada linha de G é um vetor de $\mathcal{C}$

$$HG^T = 0. \tag{4.2}$$

Peso de Hamming e Distância mínima

O peso de Hamming $w(x)$ de um vetor x é definido como o número de elementos não nulos desse vetor. No caso de existirem dois vetores, x e y, com o mesmo comprimento, a distância de Hamming $d(x, y)$ entre eles é o número de coordenadas em que estes diferem. Assim, a distância mínima $d_{\min}$ de um código $\mathcal{C}$ é o mínimo das distâncias entre quaisquer duas palavras de código diferentes. Para códigos lineares $\mathcal{C}$, a distância mínima é igual ao mínimo entre os pesos das palavras de código não nulas.

Seja $c_i \in \mathcal{C}$, uma palavra de código enviada através de um canal ruidoso. Seja v o vetor recebido ao qual pode ter sido adicionado um erro e tal que $v_i = c_i + e$. Então só podemos ter a certeza que este será corretamente corrigido se,

$$d(v_i, c_i) \leq \frac{d_{\min}(\mathcal{C}) - 1}{2}. \tag{4.3}$$

Geralmente, em códigos, utiliza-se a relação

$$d_{\min}(\mathcal{C}) \geq 2t + 1, \tag{4.4}$$

para denotar que um código corrige corretamente, com 100% certeza, até t erros. Vamos supor que recebemos uma mensagem codificada que nos foi enviada por um canal ruidoso. Como exemplo, observemos a figura 4.1. Nesta, as palavras de código são denotadas por c_i e os vetores recebidos são denotados por v_i onde o índice i relaciona a palavra de código enviada com o vetor recebido. Na parte a) temos um código cuja distância mínima não respeita a condição em 4.4 o que implica que, por exemplo, o vetor v_1, originado pelo ruído do canal de transmissão que adicionou t erros à palavra de código c_1, poderá tanto ser descodificado por c_2 como por c_1, que estão à mesma distância, o que significa que em metade dos casos este vetor v_1 seria mal corrigido. Deste modo, temos que o código em a) não corrige corretamente, com 100% certeza, até t erros, pois, o método de descodificação não é injetivo, ou seja, existe pelo menos um vetor v que apesar de possuir um erro cujo peso é igual a t, pode ser descodificado por mais que uma palavra de código. Já no caso da figura 4.1 b), o código verifica a condição 4.4 logo este consegue corrigir corretamente, e com 100% de certeza, até t erros. Caso a distância entre v_i qualquer palavra de código, seja superior a t, o descodificador irá ter dificuldade em decidir qual palavra de código deverá utilizar na descodificação optando assim, com maior probabilidade, pela palavra de código mais próxima, o que poderá levar a uma correção incorreta originando uma mensagem que não corresponde à enviada. No entanto, se o número de erros adicionados pelo canal for superior a t mas existir uma palavra de código c cuja distância entre v e c é menor ou igual a t então, este vetor será, com 100% certeza, descodificado com essa palavra de código originando também uma mensagem errada. Um exemplo destes pode ser observado na figura

4.1 b) se assumirmos que a palavra de código enviada foi c_3 e que recebemos o vetor v_3 então, como a distância entre este e c_4 é inferior a t, neste caso este será descodificado como sendo c_4.

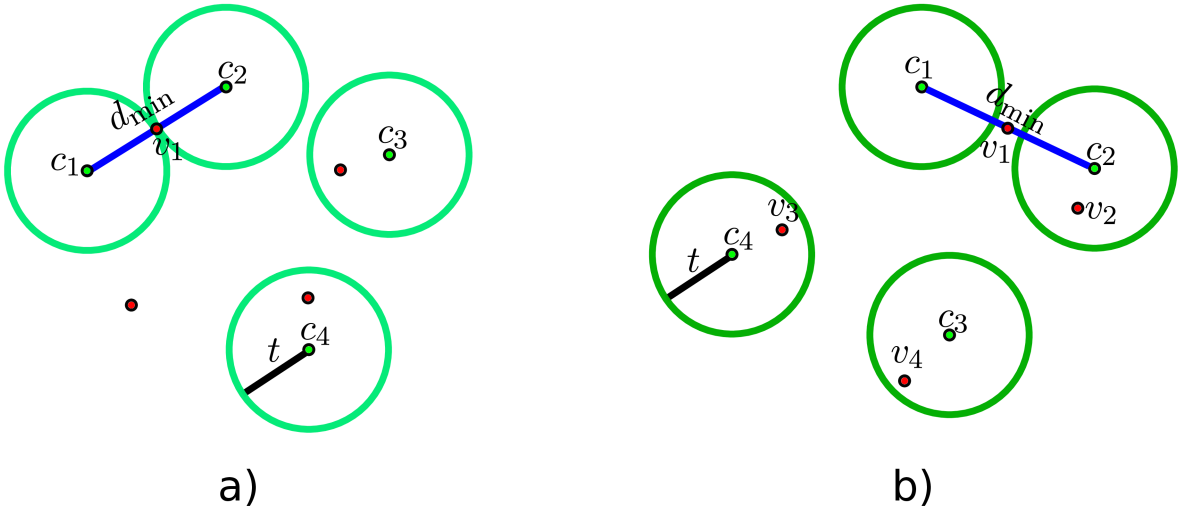


Figura 4.1: Exemplo de dois códigos: o de a) tem distância mínima inferior a $2t + 1$ e o de b) tem distância mínima maior ou igual que $2t + 1$.

4.1.2 Descodificação por síndrome

Seja $\mathcal{C}$ um código linear e c uma palavra de código em $\mathcal{C}$. Seja H a matriz paridade que representa esse código. Então, como já visto anteriormente, para qualquer palavra de código c teremos que $Hc^T = 0$. Então, se recebermos um vetor v podemos testá-lo para sabermos se ele contém ou não erros. Se $Hv^T = 0$ implica que v não tem erros o que significa que este é uma palavra de código. Se $Hv^T \neq 0$ então $v = c + e$ onde e representa o vetor erro adicionado a c.

Ao vetor $s(v) = Hv^T$ chamamos de *síndrome de v*. Esta depende do vetor erro adicionado. Deste modo é possível fazer uma correspondência direta entre cada síndrome e respetivo vetor erro com peso menor ou igual a t, onde t representa o número máximo de erros que o código pode corrigir. Como exemplo podemos construir um algoritmo simples de descodificação usando a síndrome para um código de Hamming [7:4], onde todas as colunas da sua matriz paridade são diferentes. Para isso vamos supor que temos uma mensagem $v = c + e$ que queremos descodificar. Vamos supor que $w(e) \leq 1$. Então o algoritmo consistirá no seguinte:

- pegar no vetor v que queremos descodificar e calcular a sua síndrome $s(v) = Hv^T$;
- consultar uma tabela previamente criada onde exista a correspondência entre cada possível síndrome e respetivo vetor erro;
- subtrair o respetivo vetor erro e a v obtendo assim a palavra de código c.

Esta tabela com a correspondência entre cada síndrome e cada vetor erro e pode ser obtida calculando $s(e) = He^T$ para cada possível e.

4.1.3 Códigos Reed-Solomon

Em junho de 1960, é publicado um artigo de Irving Reed e Gus Solomon, onde é descrita uma nova classe de códigos corretores de erros (ver [34]). Esses códigos, conhecidos como códigos Reed-Solomon (códigos RS), tiveram como primeira importante utilização codificar as transmissões da nave espacial Voyager, lançada na década de 70. Depois disso já adquiriram muitas mais aplicações desde serem utilizados na codificação de informação de CDs, DVDs ou video jogos.

Os códigos RS são definidos sobre um corpo $GF(q)$ e possuem tamanho $n = q - 1$. Uma matriz geradora, G, de tamanho $k \times n$, pode ser construída da seguinte forma:

$$G = \begin{bmatrix} 1 & 1 & \dots & 1 \\ \alpha_1 & \alpha_2 & \dots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \dots & \alpha_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \dots & \alpha_n^{k-1} \end{bmatrix},$$

onde $\alpha_1, \alpha_2, \dots, \alpha_n$ são os elementos não nulos de $GF(q)$. Estes, são os códigos que até ao momento possuem maior capacidade corretora de erros, ou seja, possuem maior distância d dada por $d = n-k+1$. Deste modo conseguem corrigir $t = \frac{n-k}{2}$ erros.

Analogamente podemos construir um código RS sobre o corpo $GF(q)$ (se $q = p^r$, então $GF(q)$ é o quociente $\mathbb{Z}_p[x]/(f(x))$, onde $f(x)$ é um polinómio primitivo de grau r). Como exemplo vamos considerar o corpo $GF(8)$ e o polinómio primitivo sobre este corpo $f(x) = x^3 + x + 1$. Como já visto anteriormente, os elementos possíveis para este corpo na forma de potência são $\{0, 1, x, x^2, x^3, x^4, x^5, x^6\}$. Considerando como exemplo $k = 3$, teríamos a seguinte matriz de código

$$G = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ x & x^2 & x^3 & x^4 & x^5 & x^6 & x^7 \\ x^2 & (x^2)^2 & (x^3)^2 & (x^4)^2 & (x^5)^2 & (x^6)^2 & (x^7)^2 \end{bmatrix}$$

que módulo $f(x)$ ficaria

$$G = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ x & x^2 & x+1 & x^2+x & x^2+x+1 & x^2+1 & 1 \\ x^2 & x^2+x & x^2+1 & x & x+1 & x^2+x+1 & 1 \end{bmatrix}.$$

A matriz paridade para estes códigos sobre um corpo finito do tipo $GF(2^m)$, ou seja, corpo finito de característica $p = 2$, pode ser construida da seguinte forma:

$$H = \begin{bmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & \dots & \alpha^{n-1} \\ 1 & \alpha^2 & (\alpha^2)^2 & (\alpha^2)^3 & \dots & (\alpha^2)^{n-1} \\ 1 & \alpha^3 & (\alpha^3)^2 & (\alpha^3)^3 & \dots & (\alpha^3)^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{2t} & (\alpha^{2t})^2 & (\alpha^{2t})^3 & \dots & (\alpha^{2t})^{n-1} \end{bmatrix},$$

onde $\alpha = x$ é o elemento primitivo do corpo.

Apesar destes códigos serem os que possuem maior distância, ou seja, conseguem corrigir mais erros para um tamanho de chave menor, estes não são muito utilizados em criptografia, nomeadamente no McEliece PKC, pois, são códigos que apresentam muita estrutura tornando-se assim suscetíveis a muitas classes de ataques estruturais.

4.1.4 Códigos Convolucionais

Existem várias formas de definir os códigos convolucionais. Nesta dissertação, iremos introduzir o conceito de código convolucional pela abordagem duma matriz geradora (ver [35]). Aqui iremos introduzir o conceito de *operador de delay* que denotaremos por D e que servirá para indicar o instante em que cada parte de informação u_i foi enviada ou recebida. Assim, podemos representar a sequência de vetores $u_i \in \mathbb{F}^k$ da mensagem como uma série de Laurent $u(D, D^{-1}) = u_\lambda D^\lambda + u_{\lambda+1} D^{\lambda+1} + \ldots \in \mathbb{F}^k((D))$, para todo o $\lambda \in \mathbb{Z}$. Este processo de codificação usando códigos convolucionais ficará claro mais à frente. Agora, como já referido, vamos introduzir o conceito de código convolucional.

Um código convolucional $\mathcal{C}$ de rácio k/n é um sub-espaço $\mathbb{F}((D))$ de $\mathbb{F}^n((D))$ de dimensão k dado por uma matriz, racional, codificadora $G(D) \in \mathbb{F}^{k\times n}(D)$,

$$\mathcal{C} = \mathrm{Im}_{\mathbb{F}((D))} G(D) = \{u(D, D^{-1})G(D) : u(D, D^{-1}) \in \mathbb{F}((D))\}, \tag{4.5}$$

onde $u(D, D^{-1}) = \sum_{i \geq \lambda} u_i D^i$ é chamado de vetor de informação. Se

$$G(D) = \sum_{i=0}^{m} G_i D^i \in \mathbb{F}^{k\times n}[D] \tag{4.6}$$

é polinomial, m é denominado por *memória* de $G(D)$, pois, este necessita de ter armazenados os valores de entrada u_i dos m instantes anteriores. Podemos notar que quando $m = 0$, o codificador é constante originando assim um código de bloco. Logo, a classe dos códigos convolucionais pode ser vista como uma generalização da classe dos códigos lineares de bloco.

Uma descrição dual de um código convolucional $\mathcal{C}$ pode ser dada por uma das suas matrizes paridade $H(D)$, de dimensão $(n-k) \times n$ e rácio $(n-k)/n$, tal que

$$\mathcal{C} = \ker_{\mathbb{F}((D))} H(D) = \{v(D) \in \mathbb{F}((D)) | H(D)v(D) = 0 \in \mathbb{F}^{n-k}((D))\}. \tag{4.7}$$

4.2 Criptossistema de McEliece

O criptossistema de McEliece é um criptossistema de chave pública, PKC baseado em códigos, proposto por Robert J. McEliece em 1978. Este criptossistema assentava no facto de existir um algoritmo rápido para descodificar códigos Goppa enquanto que o mesmo não se verificava para a generalidade dos códigos (ver [9]). Deste modo, a sua chave privada é uma matriz geradora de código Goppa, binário, irredutível e aleatório, enquanto que a sua chave pública é uma matriz geradora que aparenta ser aleatória, obtida através da multiplicação da chave privada por uma matriz invertível e uma permutação. À mensagem codificada será adicionado um erro, de peso t, de acordo com a capacidade corretora do código. Assim, apenas quem possuir a chave privada será capaz de remover esses erros e obter a mensagem original.

4.2.1 Descrição

Inicialmente, Bob gera 3 matrizes, S, G e P, que constituirão a chave privada. A matriz G é uma matriz geradora de código de tamanho $k \times n$ que corrige $t \leq \frac{d-1}{2}$ erros, a matriz S é invertível e aleatória do tipo $k \times k$ e a matriz P é uma matriz de permutação do tipo $n \times n$. De seguida, Bob computa $G' = SGP$ e publica (G', t) como sendo a sua chave pública. Note-se que, no caso original, a matriz G seria um irredutível código Goppa binário.

Vamos supor que o Alice quer cifrar uma mensagem binária, u, e enviar para o Bob. Então, Alice:

- Divide a sua mensagem em i blocos de tamanho k, ficando assim com $u = (\overbrace{u_0, u_1, \ldots, u_{i-2}, u_{i-1}}^{i})$;
- Multiplica cada um dos blocos por G' e depois soma a cada um deles um vetor e_j (onde $j = 0, 1, \ldots, i-2, i-1$), de comprimento n e peso máximo t, que é gerado por ela aleatoriamente e mantido secreto. Deste modo, a sua mensagem codificada será um vetor $y = [y_0, y_1, \ldots, y_{i-2}, y_{i-1}]$ onde $y_j = u_j G' + e_j$.

Para decifrar a mensagem, y, enviada pela Alice, o Bob segue os seguintes passos:

- Computa $y'_j = y_j P^{-1}$ onde P^{-1} é a matriz inversa de P. Deste modo terá, $y'_j = (u_j G' + e_j)P^{-1} = (u_j SGP + e_j)P^{-1} = u_j SG + e_j P^{-1}$. Como P é uma matriz permutação, a sua inversa será também uma matriz permutação e como tal, o peso de cada vetor e não será alterado após a multiplicação por esta;
- Como o código escolhido admite um algoritmo de descodificação fácil, Bob faz a descodificação e obtém $u'_j = u_j S$;
- Finalmente, multiplica o resultado do ponto anterior, u'_j, por S^{-1} e obtém $u'_j S^{-1} = u_j S S^{-1} = u_j$.

4.3 Limitações e Ataques

Comecemos por definir Fator de Trabalho, *Work Factor* (WF). O WF de um criptossistema é uma medida relacionada com o método utilizado por quem tenta quebrar o sistema, juntamente com o número de operações necessárias e o custo de cada uma delas. O resultado desta medida é um número médio de iterações necessárias pelo custo de cada uma delas, para que, segundo um

determinado algoritmo de ataque, se consiga quebrar o sistema. Deste modo, quando maior o WF de um criptossistema, mais seguro ele será contra o método de ataque para o qual este foi calculado. Por exemplo, se pensarmos num ataque à força bruta onde teríamos de tentar todas as combinações possíveis da chave, facilmente percebemos que neste caso o WF seria proporcional ao tamanho da chave. No entanto, não se utilizam métodos de força bruta mas sim outros de maior complexidade mas que diminuem em várias ordens de grandeza este WF. Atualmente, a classe de ataques não estruturais mais eficazes contra os criptossistemas baseados em códigos, é a Descodificação por Conjunto de Informação, ISD [36]. Portanto, todos os valores de WF mencionados serão relativos a esta classe de ataques.

Assim, o McEliece PKC original apresenta várias desvantagens. Uma delas, e a mais importante, é o facto de que para termos um WF de pelo menos 2^{128}, ou seja, garantir uma segurança de 128 bits, precisaríamos de um código Goppa binário de tamanho $(2960, 2288)$ gerado sobre um corpo finito com 2048 elementos ($GF(2048)$). Isto resultaria num tamanho de chave pública de 1537536 bits (ver [37]), o que não é prático.

Tendo em vista reduzir o tamanho da chave sem comprometer a segurança do sistema, outras variantes foram posteriormente propostas utilizando outros códigos que não os Goppa e\ou outras formas de esconder o código G por trás, sem aumentarem o tamanho da chave pública e garantindo na mesma um elevado nível de segurança.

Uma dessas novas propostas para o McEliece PKC foi desenvolvida por nós. Esta consiste no uso de códigos convolucionais (ver [11]) e será apresentada um pouco mais à frente.

4.3.1 Descodificação por Conjunto de Informação

A descodificação por conjunto de informação, ISD, é considerada a melhor classe de ataques não estruturais contra o McEliece onde se procura obter logo a mensagem original através da mensagem cifrada e da chave pública em vez de se tentar obter a chave privada para decifrar a mensagem.

Comecemos por definir conjunto de informação $I \subset \{1, \ldots, n\}$ como uma sequência de tamanho k que indexa k colunas linearmente independentes de G' onde, G' é uma matriz geradora de código sobre um código linear $\mathbb{F}_q$ de dimensão $k \times n$ [36].

O objetivo desta classe de ataques (ISD) é, dada esta matriz G' e dado um vetor $y = uG' + e$, descobrir a mensagem u. No primeiro ataque deste tipo, conhecido como algoritmo de Prange, é procurado um conjunto de informação I tal que nas posições por este indexadas, y não contenha erros. Posteriormente foram desenvolvidos outros algoritmos, nomeadamente o Lee-Brickell e o Stern. Uma ilustração geral de como o conjunto de informação é dividido nestes diferentes algoritmos encontra-se representada na figura 4.2. Destes o mais utilizado é o Stern onde a principal diferença está distribuição dos erros, ou seja, dado uma palavra $y \in \mathbb{F}_q^n$ cuja distância ao código $\mathbb{F}_q^n G$ é w, é escolhido um conjunto de informação I e divide-se em duas partes iguais I_1 e I_2 de tamanho $n - k$, onde se supõe que em cada uma existe p erros. As restantes $n-k$ colunas de y também são divididas em duas partes onde se assume que na primeira, de tamanho l, existe zero erros. Deste modo temos um vetor de distribuição de erros, representado na figura 4.2.

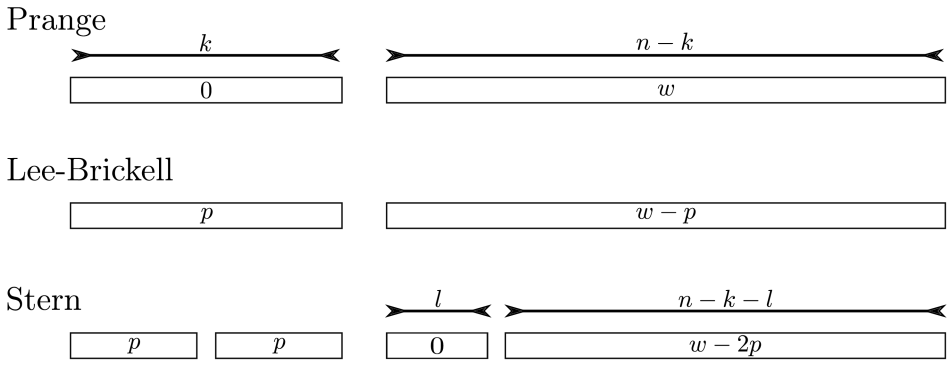


Figura 4.2: Alguns algoritmos da classe de ataques ISD e respetiva distribuição de erros.

4.4 Nova variante do McEliece PKC baseada em códigos convolucionais

Nesta secção iremos descrever uma nova variante do McEliece PKC [11] e, de seguida, dar alguns exemplos numéricos com o respetivo cálculo do fator de trabalho, terminando com um exemplo prático.

4.4.1 Descrição

De forma a percebermos melhor esta nova variante do McEliece PKC, vamos dividir a sua descrição em 3 partes: gerar a chave, cifrar e decifrar.

1. Gerar a chave

   Tal como na abordagem do McEliece PKC original, vamos supor que é o Bob quem gera e publica a chave. Então Bob,

   - Gera uma matriz de um código de bloco $G \in \mathbb{F}^{k\times n}$, que admita um algoritmo de descodificação fácil e capaz de corrigir t erros.
   - De seguida gera
   $$T(D, D^{-1}) = \sum_{i=-\mu}^{\mu} T_i D^i \in \mathbb{F}^{n\times n}(D, D^{-1}) \tag{4.8}$$
   uma matriz polinomial de Laurent, invertível em $\mathbb{F}(D, D^{-1})$, tal que, o seu determinante pertence a $\mathbb{F}$, as posições das colunas não nulas de T_i formam uma partição de n e cada linha de T_i tem pelo menos um elemento não nulo para $i = -\mu, \ldots, 0, \ldots, \mu$.
   - Por último gera
   $$S(D) = \sum_{i=\mu}^{\upsilon} S_i D^i \in \mathbb{F}^{k\times k}[D] \tag{4.9}$$
   com $S_\mu \in \mathbb{F}^{k\times k}$, uma matriz invertível.

   Como a matriz $T(D, D^{-1})$ é invertível, ele inverte-a obtendo $P(D, D^{-1}) = T^{-1}(D, D^{-1})$. Deste modo a sua chave privada será $\{S(D), G, P(D, D^{-1})\}$. De seguida ele calcula
   $$G'(D) = S(D)GP(D, D^{-1}) \tag{4.10}$$
   e publica $\{G'(D), t, \mu\}$ como sendo a chave pública.

   Pelo corolário 2.2 em [35], como o inverso de $T(D, D^{-1})$ é
   $$P(D, D^{-1}) = \sum_{i=-\mu}^{\mu} P_i D^i \in \mathbb{F}^{n\times n}(D, D^{-1}), \tag{4.11}$$
   onde as posições das linhas não nulas de P_i formam uma partição de n. Então, $G'(D)$ como em 4.10 é polinomial e tem memória $m \leq \mu + \upsilon$. A prova deste corolário também pode ser encontrada em [35].

2. Cifrar

   Vamos supor que a Alice quer enviar uma mensagem cifrada para o Bob. Seja a mensagem uma sequência $(u_0, u_1, \cdots)$ onde $u_i \in \mathbb{F}^k$ e é representado como um polinómio $u(D) = u_0 + u_1 D + \ldots + u_l D^l \in \mathbb{F}^k((D))$. Então a Alice,

   - Seleciona um vetor aleatório $e(D) = \sum_{i\geq 0} e_i D^i \in \mathbb{F}^n[D]$ tal que $w((e_i, e_{i+1}, \ldots, e_{i+2\mu})) \leq t$ para todo $i \geq 0$.
   - Cifra $u(D)$ como
   $$y(D) = u(D)G'(D) + e(D). \tag{4.12}$$

Ou seja, vamos considerar então $m = \mu + \upsilon$. Como cada u_i é um vetor de tamanho k, podemos ver este último passo da encriptação como, numa primeira fase, a codificação de $u(D)$ por $G'(D)$ fazendo

$$\begin{bmatrix} u_0 & u_1 & \dots & u_l \end{bmatrix} \begin{bmatrix} G'_0 & G'_1 & \dots & G'_m & & & & \\ & G'_0 & G'_1 & \dots & G'_m & & & \\ & & \ddots & \ddots & & \ddots & & \\ & & & G'_0 & G'_1 & \dots & G'_m & \\ & & & & \ddots & \ddots & & \ddots \\ & & & & & G'_0 & G'_1 & \dots & G'_m \end{bmatrix}$$

e por último, a soma do resultado anterior com o vetor

$$\begin{bmatrix} e_0 & e_1 & \dots & e_{l+m} \end{bmatrix}$$

obtendo assim

$$\begin{bmatrix} y_0 & y_1 & \dots & y_{l+m} \end{bmatrix}.$$

3. Decifrar

   Bob, como portador da chave privada, pega na mensagem $y(D)$ enviada pela Alice e:

   - Multiplica pela direita por $T(D, D^{-1})$, ou seja, faz

   $$\begin{bmatrix} y_0 & y_1 & \dots & y_{l+m} \end{bmatrix} \begin{bmatrix} T_{-\mu} & \dots & T_0 & \dots & T_\mu & & & & \\ & T_{-\mu} & \dots & T_0 & \dots & T_\mu & & & \\ & & \ddots & & \ddots & & \ddots & & \\ & & & T_{-\mu} & \dots & T_0 & \dots & T_\mu & \\ & & & & \ddots & & \ddots & & \ddots \\ & & & & & T_{-\mu} & \dots & T_0 & \dots & T_\mu \end{bmatrix}$$

   obtendo

   $$y(D)T(D, D^{-1}) = u(D)S(D)GP(D, D^{-1})T(D, D^{-1}) + e(D)T(D, D^{-1}). \tag{4.13}$$

   Como $T(D, D^{-1})$ é a inversa de $P(D, D^{-1})$, Bob após esta operação fica com $u(D)S(D)G + e(D)T(D, D^{-1})$;

   - Usa um algoritmo de descodificação para G ficando com $u(D)S(D)$;
   - Para o último passo, consideremos

   $$u(D)S(D) = \sum_{i=\mu}^{l+\upsilon} \widehat{u}_i D^i. \tag{4.14}$$

   Como $\begin{bmatrix} \widehat{u}_\mu & \widehat{u}_{\mu+1} & \dots & \widehat{u}_{l+\upsilon} \end{bmatrix} =$

   $$= \begin{bmatrix} u_0 & u_1 & \dots & u_l \end{bmatrix} \underbrace{\begin{bmatrix} S_\mu & S_{\mu+1} & \dots & S_\upsilon & & & \\ & S_\mu & S_{\mu+1} & \dots & S_\upsilon & & \\ & & \ddots & \ddots & \ddots & \ddots & \\ & & & S_\mu & \dots & & S_\upsilon \\ & & & & \ddots & & \vdots \\ & & & & & \ddots & \vdots \\ & & & & & & S_\mu \end{bmatrix}}_{=:S_{truc}(l) \in \mathbb{F}^{(l+\lambda+1)k \times (l+\lambda+1)k}},$$

torna-se fácil visualizar que

$$\begin{aligned} \widehat{u}_{\mu} &= u_0 S_{\mu} && (4.15)\\ \widehat{u}_{\mu+1} &= u_0 S_{\mu+1} + u_1 S_{\mu} && (4.16)\\ &\vdots \end{aligned}$$

Assim, como S_μ é invertível, Bob pode facilmente obter toda a mensagem original começando por fazer

$$\begin{aligned} u_0 &= \widehat{u}_{\mu} S_{\mu}^{-1} \\ &= u_0 S_{\mu} S_{\mu}^{-1} && (4.17)\\ u_1 &= (\widehat{u}_{\mu+1} - u_0 S_{\mu+1}) S_{\mu}^{-1} && (4.18)\\ &\vdots \end{aligned}$$

e terminando em u_l.

Como G apenas pode corrigir t erros, nós tivemos que garantir que, em cada instante, $e(D)T(D, D^{-1})$ tem peso menor ou igual que t, ou seja, considerando que $e(D) = \sum_{i\geq 0} e_i D^i \in \mathbb{F}^n[D]$ garantir que

$$w((e_i, e_{i+1}, \ldots, e_{i+2\mu})) \leq t \tag{4.19}$$

para todo $i \geq 0$. Isto foi possível construindo a matriz $T(D, D^{-1})$ com aquela estrutura particular, pois, de acordo com o lema 2.3 em [35], se tivermos $T(D, D^{-1})$ como descrito anteriormente e se $e(D) = \sum_{i\geq 0} e_i D^i \in \mathbb{F}^n[D]$ é um vetor aleatório que satisfaz 4.19 para todo $i \geq 0$, então, todos os coeficientes $e(D)T(D, D^{-1})$ tem peso menor ou igual que t. A prova a este lema também pode ser encontrada em [35].

A principal ideia deste esquema é mascarar uma matriz geradora de código G, de baixo grau e algoritmo de descodificação rápido, e apresentar como chave pública uma matriz $G'(D)$, de grau elevado, cuja descodificação é impraticável por quem não possui as componentes privadas.

4.4.2 Segurança e exemplos numéricos

Há duas principais classes de ataques contra o criptossistema de McEliece. Uma delas consiste em atacar a estrutura do código e a outra, é a classe dos ISD. Como na nossa variante, $G'(D)$ é construido através de matrizes geradas maioritariamente de forma aleatória, isto faz com que a estrutura de G fique suficientemente mascarada o que torna o tipo de ataques estruturais mais difícil. Deste modo vamos focarmo-nos apenas nos ataques ISD.

Se um atacante a este criptossistema usar um ISD, devemos considerar duas diferentes possibilidades: ou ele faz um ataque global, onde utiliza a mensagem toda, ou faz um ataque local onde usa apenas alguns intervalos da mensagem. Vamos começar por analisar o caso do ataque global. Neste caso a matriz

$$\begin{bmatrix} G'_0 & G'_1 & \ldots & G'_m & & & & & \\ & G'_0 & G'_1 & \ldots & G'_m & & & & \\ & & \ddots & \ddots & & \ddots & & & \\ & & & G'_0 & G'_1 & \ldots & G'_m & & \\ & & & & \ddots & \ddots & & \ddots & \\ & & & & & G'_0 & G'_1 & \ldots & G'_m \end{bmatrix},$$

geradora de um código de bloco, é toda utilizada. No entanto, como o tamanho deste código bloco é $k(l+1)\times n(l+1+\mu+\upsilon)$, basta aumentarmos os valores de l e υ para aumentarmos o valor do WF para este ataque, tornando-o inviável. Já para o caso de um ataque local, o atacante vai considerar apenas uma sequência da mensagem $y(D)$. No entanto, como estamos a usar códigos convolucionais, cada y_i depende de alguns y anteriores o que implica que para descobrirmos o estado do código convolucional

no instante i precisamos também de conhecer alguns instantes antes. Logo, não faz sentido ele pegar numa sequência em $y(D)$ que não comece em y_0. Deste modo vamos assumir que ele vai tentar atacar os primeiros vetores $[y_0 y_1 \dots y_s]$ para diferentes valores de s, truncando desta forma a matriz geradora do código bloco como $G'_{\text{truc}}(s)$, ou seja, ficando

$$\begin{bmatrix} u_0 & u_1 & \dots & u_s \end{bmatrix} \underbrace{\begin{bmatrix} G'_0 & G'_1 & \dots & G'_s \\ & G'_0 & \dots & G'_{s-1} \\ & & \ddots & \vdots \\ & & & G'_0 \end{bmatrix}}_{=:G'_{truc}(s)} + \begin{bmatrix} e_0 & e_1 & \dots & e_s \end{bmatrix} = \begin{bmatrix} y_0 & y_1 & \dots & y_s \end{bmatrix},$$

onde $s \leq l$ e $G'_i = 0$ para $i > m$. Vamos supor que $G'(D)$ foi truncada em $s = m$. Deste modo iremos considerar ataques da classe ISD à matriz $G'_{\text{truc}}(s)$ e iremos calcular o fator de trabalho destes ataques. Para isso, vamos dividir o cálculo do WF em duas partes:

1. parte clássica calculada de acordo com [36] que denotaremos por $\text{WF}_{\text{classico}}(n_s, k_s, q, t_s)$,
2. parte característica desta variante que está relacionada com facto de existir ainda $k(s+1) - k_s$ linhas que não são linearmente independentes o que implica que, para elas, teremos que testar todas as possibilidades de erros, ou seja, testar $q^{k(s+1)-k_s}$ combinações.

Em cima, n_s representa o comprimento de $G'_{\text{truc}}(s)$ e k_s representa o número de linhas linearmente independentes, que é a *característica* (*rank*), de $G'_{\text{truc}}(s)$, e que denotaremos por $r(G'_{\text{truc}}(s))$, t_s representa o número máximo de erros no intervalo $[0 : s]$, q é o tamanho do corpo e k é a dimensão de $G'(D)$. Assim, o nosso fator de trabalho total é dado por

$$\text{WF}_{\text{conv}}(s, k, n_s, k_s, q, t_s) = q^{k(s+1)-k_s} \text{WF}_{\text{classico}}(n_s, k_s, q, t_s). \tag{4.20}$$

No entanto, pela forma como as matrizes da chave privada são geradas, nós não conseguimos definir um valor certo para k_s por isso o que iremos aqui fazer é estimar esse valor para o pior cenário possível de WF. Assim, consideremos um código convolucional $\mathcal{C}$ de rácio k/n com codificador $G(D)$ e memória m. Seja k_s igual $r(G'_{\text{truc}}(s))$. Então, para esta nova variante do McEliece, o WF para um ataque local do tipo ISD a $G'_{\text{truc}}(s)$ terá um limite inferior dado por

$$\text{WF}(n(s+1), k(s+1), q, t_s) \geq q^{k(s+1)-k_s} \text{WF}_{\text{classico}}(n(s+1), k_s, q, t_s). \tag{4.21}$$

Notemos que, como estamos a considerar $s = m$ então $n_s = n(s+1)$. Podemos verificar que $\text{WF}_{\text{classico}}(n(s+1), k_s, q, t_s)$ é crescente com o aumento de k_s enquanto que $q^{k(s+1)-k_s}$ é decrescente com esse mesmo aumento. Logo, para estimarmos o valor mais baixo que o nosso WF poderá ter, vamos estimar um valor máximo para o k_s do WF_{conv} que denotaremos por $\overline{k}$ e um valor mínimo de k_s para o $\text{WF}_{\text{classico}}$ que denotaremos por $\underline{k}$. Assim podemos reescrever 4.21 como

$$\text{WF}(n(s+1), k(s+1), q, t_s) \geq q^{k(s+1)-\overline{k}} \text{WF}_{\text{classico}}(n(s+1), \underline{k}, q, t_s). \tag{4.22}$$

Uma estimativa para estes valores seria tomar $\overline{k} = k \cdot s \cdot r(G'_0)$ e $\underline{k} = s \cdot r(G'_1)$. Para os exemplos futuros onde calcularemos o WF para matrizes deste tipo (truncadas), este será calculado tendo em conta estas estimativas.

Seja o número máximo de erros que podem ser corrigidos, satisfazendo a equação (4.19), dados por

$$\left\lfloor \frac{t(l+m)}{2\mu + 1} \right\rfloor, \tag{4.23}$$

onde t é a capacidade corretora de G e $m = \mu + \upsilon$. Seja o tamanho da chave privada dado por

$$r \cdot \lceil \log_2 p \rceil \cdot n \cdot k(m+1), \tag{4.24}$$

onde a ordem do corpo é $q = p^r$ com p primo. Observemos agora os seguintes exemplos:

- **Exemplo 1.** Consideremos os parâmetros $n = 63$, $k = 45$, $m = \mu + \upsilon = 3 + 16 = 19$ e $q = 2^6$. Consideremos também que G é gerada a partir de um código RS e que a mensagem/chave

$u_0u_1 \dots u_l$ a ser enviada tem 4500 bits. Isto implica que $l = \frac{4500}{k} = 100$ e que podem ser corrigidos $\lfloor \frac{9(l+19)}{3} \rfloor = 357$ erros. Logo, o WF para um ataque global é 2^{501};

- **Exemplo 2.** Consideremos os parâmetros $n = 63$, $k = 45$, $m = \mu + \upsilon = 3 + 16 = 19$ e $q = 2^6$. Consideremos também que G é gerada a partir de um código RS e que a matriz bloco G' é truncada em $s = m = 19$. Isto implica que podem ser corrigidos $\lfloor \frac{180}{3} \rfloor = 60$ erros. Então o WF para um ataque ISD a este criptossistema seria 2^{301}. O tamanho da chave privada para este exemplo é 340200 bits.

Na tabela seguinte temos a comparação entre o tamanho da chave privada e o WF da nossa proposta, apresentada no exemplo anterior (exemplo 2.), e os do criptossistema proposto por Niederreiter para uma segurança de 128 bits.

	Goppa binário	Nossa proposta usando códigos RS
k	2288	45
n	2960	63
q	2048	64
m	0	19
WF	2^{128}	2^{301}
tamanho da chave (bits)	1537536	351000

Tabela 4.3: Comparação entre o McEliece PKC de Niederreiter e a nossa proposta.

Podemos, para este exemplo, verificar que para uma chave com 4 vezes menos bits que a chave proposta por Niederreiter, apresentamos um WF muito maior. Assim, esta variante do McEliece descrita nesta dissertação, apresenta para um tamanho de chave equivalente às outras variantes, uma segurança maior.

4.4.3 Exemplo prático

Consideremos a seguinte matriz geradora para o código de Hamming [38] $\mathcal{C} - [7, 4]$ sobre $GF(2)$, capaz de corrigir até $t = 1$ erros,

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

e respetiva matriz paridade

$$H = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Consideremos também as seguintes matrizes $S(D)$ e $T(D, D^{-1})$, geradas com as características previamente enunciadas, para os parâmetros $\upsilon = 5$ e $\mu = 1$

$$S(D) = \begin{bmatrix} D & D^5 + D^4 + D^3 & D^4 & D^5 \\ 0 & 0 & D^2 & D \\ 0 & D & 0 & 0 \\ 0 & 0 & D & 0 \end{bmatrix},$$

$$T(D, D^{-1}) = \begin{bmatrix} D & 0 & D^{-1} & 0 & 0 & 0 & 0 \\ 0 & D & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ D & 0 & 0 & D^{-1} & 0 & 0 & 0 \\ 0 & D & 0 & D^{-1} & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & D & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Apenas Bob possui estes dados sendo as matrizes anteriores a sua chave privada. Com estas, Bob começa por calcular $P(D, D^{-1}) = T^{-1}(D, D^{-1})$, obtendo

$$P(D, D^{-1}) = \begin{bmatrix} 0 & 0 & 0 & D^{-1} & -D^{-1} & D^{-1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{D} \\ D & 0 & 0 & -D & D & -D & 0 \\ 0 & 0 & 0 & -D & D & -D & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

De seguida calcula

$$G'(D) = S(D)GP(D, D^{-1}) = \begin{bmatrix} D^5 & D^3 & D^4 + D^3 + D & D^5 + 1 & D^6 + D^5 + 1 & D^6 + D^4 + D + 1 & D^5 + D^2 + D \\ D^3 & D^2 + D & D & D^3 & D^3 + D^2 & D^3 + D & 0 \\ 0 & D & D & 0 & 0 & 0 & D + 1 \\ D^2 & D & 0 & D^2 & D^2 & D^2 + D & 0 \end{bmatrix}.$$

e publica como chave privada $(G'(D), t, \mu)$.

Vamos supor que Alice quer cifrar a mensagem *Criptografia* e enviá-la para o Bob. Então, ela primeiro converte a mensagem para binário ficando com
u =[0 1 0 0 0 0 1 1 0 1 1 1 0 0 1 0 0 1 1 0 1 0 0 1 0 1 1 1 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 0 1 1 1 1 0 1 1 0
0 1 1 1 0 1 1 1 0 0 1 0 0 1 1 0 0 0 0 1 0 1 1 0 0 1 1 0 0 1 1 0 1 0 0 1 0 1 1 1 0 0 0 0 1],
calcula
$c = uG'(D, D^{-1})$ =[0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 1 1 0 1 1 0 0 1 0 0 0 0 0 1 0 1 1 0 1 0 1 0 0 0 0 1 1 0
0 1 01 0 0 0 1 0 0 0 0 0 1 0 1 1 0 1 1 1 1 1 1 1 0 1 0 0 1 0 1 0 0 1 1 0 0 0 0 0 1 0 1 1 0 0 1 0 1 0 0 1 1
1 1 1 1 1 0 1 0 0 0 1 0 1 1 0 1 0 1 1 0 0 1 1 0 0 1 0 1 1 0 0 0 0 1 1 1 0 0 0 0 0 1 0 0 1 0 0 1 1 0 1 1 0 0
1 1 1 1 0 1 1 0 1 0 0 0 0 0 0 1 0 0 1 1 0 1 1 0 0 0 1 0 0 1 0 1 0 0 1 1 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0].
De seguida gera o vetor erro aleatório
e =[0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0],
calcula
$y = c + e$ =[0 0 0 0 0 1 0 0 1 1 0 0 1 1 0 1 1 0 1 1 0 0 1 0 0 0 0 0 1 0 1 1 0 1 0 1 0 0 0 0 1 1 0 0 1 0 1 0
0 0 1 0 0 0 1 0 1 0 1 1 0 1 1 1 1 1 1 1 0 1 0 0 1 0 1 0 0 1 1 0 0 0 0 0 1 0 1 1 0 0 1 0 1 0 0 1 1 1 1 1 1 0 0
1 0 0 0 1 0 1 1 0 1 0 1 1 0 0 1 1 0 0 1 0 1 1 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 1 0 0 1 1 0 1 1 0 0 1 1 1 1 0 1
1 1 1 0 0 0 0 0 0 1 0 0 1 1 0 1 1 0 0 0 1 0 0 1 0 1 0 0 1 1 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0].
e envia y para o Bob.

O Bob, para decifrar a mensagem recebida, calcula
$yT(D, D^{-1}) = uS(D)G + eT(D, D^{-1})$ =[0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 1 1 1 0 1 0 0 1 0 1 0 1 1 1
1 0 0 0 1 1 0 0 1 1 0 1 0 1 0 1 0 1 0 0 0 0 1 1 0 1 1 1 1 1 0 0 0 1 0 0 1 1 0 0 0 1 1 1 1 0 1 1 0 0 1 1 0 1
1 1 1 0 0 1 1 0 1 0 0 1 0 1 1 0 1 0 0 0 0 0 0 1 1 1 1 1 1 0 1 1 0 0 1 0 0 1 1 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0
1 1 0 1 0 1 0 0 1 1 1 1 0 0 0 1 1 0 1 0 0 1 0 0 0 0 0 1 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 1 1 0 0 1 1 0 1 0 1 0
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0].
De seguida faz a descodificação por síndrome e obtém
$uS(D)$ =[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 0 0 1 0 1 0 1 1 1 1 0 0 0 1 1 0 0 1 1 0 1 0 1 0 1 0
1 0 0 0 0 1 1 0 1 1 1 1 0 0 0 1 1 0 0 1 1 0 0 0 1 1 1 1 0 1 1 0 0 1 1 0 1 1 1 1 1 0 0 1 1 0 1 0 0 1 0 1 1 1 1
0 0 0 0 0 1 1 1 1 0 0 1 1 0 0 1 0 0 1 1 0 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 1 1 1 1 0 0 0 1 1
1 1 0 0 1 0 0 0 0 1 1 0 1 0 1 0 1 0 0 1 0 0 1 0 1 0 1 1 0 0 1 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0].
Por último utiliza o algoritmo anteriormente descrito e obtém
u =[0 1 0 0 0 0 1 1 0 1 1 1 0 0 1 0 0 1 1 0 1 0 0 1 0 1 1 1 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 0 1 1 1 1 0 1 1 0
0 1 1 1 0 1 1 1 0 0 1 0 0 1 1 0 0 0 0 1 0 1 1 0 0 1 1 0 0 1 1 0 1 0 0 1 0 1 1 1 0 0 0 0 1],
que ao converter para ASCII obtém a mensagem original *Criptografia*.

Para o exemplo, utilizamos valores baixos de μ e υ e uma matriz geradora do código de Hamming $\mathcal{C} - [7, 4]$ em vez de uma matriz geradora de um código Reed-Solomon, pois, além de tornar o exemplo mais pequeno, torna mais fácil a visualização de todo o processo de encriptação e desencriptação. Além disso, como G é pequena corrigindo no máximo 1 erro, existem poucas síndromes possíveis logo podemos usar facilmente a descodificação por síndrome.

4.5 Conclusões

Neste capítulo, apresentámos uma nova variante do criptossistema de McEliece, baseada em códigos convolucionais, que poderá servir como alternativa aos criptossistemas clássicos atuais. Este apresenta inúmeras vantagens em relação às restantes variantes, como por exemplo, para um tamanho de chave mais pequeno possui um WF muito maior. Esta nova variante do McEliece, foi totalmente implementada computacionalmente para códigos de Hamming e, até à parte de decifrar, para códigos Reed-Solomon, ou seja, para quaisquer parâmetros, o código gera todas as matrizes $(S(D), G, T(D, D^{-1}), P(D, D^{-1}))$, respetiva chave privada e cifra qualquer mensagem. Também foi criado um programa para calcular o WF (ver [13]) para quaisquer parâmetros desta, e das restantes variantes, deste criptossistema. Assim, apenas esta programação já nos exigiu, além da aprendizagem sobre o uso de linguagem simbólica e implementação de corpos finitos em python, uma boa aprendizagem sobre teoria dos códigos. Apesar dos bons resultados que obtivemos com esta nova variante, iremos continuar a testar novos parâmetros e matrizes geradoras de outros tipos de códigos de modo a melhorá-la cada vez mais.

Referências

[1] M. Fox. *Quantum Optics - An Introduction*. Oxford University Press Inc., 2006. URL: global.oup.com/academic/product/quantum-optics-9780198566731.

[2] E. Rieffel e W. Polak. *Quantum Computing - A Gentle Introduction*. The MIT Press, 2011. URL: mitpress.mit.edu/books/quantum-computing.

[3] N. S. Dattani e N. Bryans. *Quantum factorization of 56153 with only 4 qubits*. 2014. arXiv: 1411.6758v3.

[4] ESTI. «Quantum Safe Cryptography and Security - An introduction, benefits, enablers and challenges». 2015. URL: https://www.etsi.org/images/files/ETSIWhitePapers/QuantumSafeWhitepaper.pdf.

[5] E. Cartlidge. «Quantum Computing: How Close Are We?» Em: *Optics & Photonics News* (out. de 2016). URL: https://www.osa-opn.org/home/articles/volume_27/october_2016/features/quantum_computing_how_close_are_we/.

[6] S. Dougal. *IBM, Google and Intel jostle for quantum computing supremacy*. 2018. URL: www.cbronline.com/news/ibm-google-intel-quantum-computing (acedido em 01/07/2018).

[7] IBM. *The intricate beauty of an IBM quantum computer*. 2017. URL: www.facebook.com/IBM/photos/a.490518774371743.1073741825.168597536563870/1521835057906771/ (acedido em 01/07/2018).

[8] T. Greene. *Google reclaims quantum computer crown with 72 qubit processor*. 2018. URL: thenextweb.com/artificial-intelligence/2018/03/06/google-reclaims-quantum-computer-crown-with-72-qubit-processor/ (acedido em 01/07/2018).

[9] R. J. McEliece. *A Public-Key Cryptosystem Based On Algebraic Coding Theory*. Rel. téc. 1978, pp. 114–116. URL: adsabs.harvard.edu/abs/1978DSNPR..44..114M.

[10] C. Löndahl e T. Johansson. «A new version of mceliece pkc based on convolutional codes». Em: *Information and Communications Security*. Ed. por I. T. W. Chim e T. H. Yuen. Springer Berlin Heidelberg, 2012, pp. 461–470. URL: link.springer.com/book/10.1007/978-3-642-34129-8.

[11] J. Brandão, C. Sebastião, P. Almeida e D. Napp. «Convolutional code-based Cryptosystem». Em: *Red Temática de Álgebra Lineal, Análisis Matricial y Aplicaciones (ALAMA)*. Universidad de Alicante, 2018, pp. 117–119. ISBN: 978-84-16724-96-3.

[12] J. Brandão. *Computational implementation of Draper adder, Quantum Fourier Transform and Quantum Shor algorithm*. https://github.com/jorgebrandao/ShorAlgorithm. 2018.

[13] J. Brandão. *Computational implementation of a new variant of McEliece PKC with convolutional codes*. https://github.com/jorgebrandao/McEliece. 2018.

[14] P. J. Almeida e D. Napp. *Criptografia e Segurança*. Publindústria, Edições Técnicas, 2017. URL: engebook.com/2/12866/Criptografia-e-Seguranca.

[15] R. A. Mollin. *Fundamental Number Theory With Applications*. Ed. por Kenneth H. Rosen. 2ª ed. Chapman e Hall/CRC, 2008. URL: www.crcpress.com/9781420066593.

[16] P. Ribenboim. *The New Book of Prime Number Records*. 3rd. Springer-Verlag, 1996. URL: springer.com/gp/book/9780387944579.

[17] R. A. Mollin. *RSA and Public-Key Cryptography*. CRC Press LLC, 2003. URL: crcpress.com/9781584883388.

[18] L. Fortnow. «The Status of the P versus NP Problem». Em: *Communications of the acm* (set. de 2009). DOI: 10.1145/1562164.1562186.

[19] M. A. Nielsen e I. L. Chuang. *Quantum Computation and Quantum Information*. 10th Anniv. Cambridge University Press, 2010. URL: cambridge.org/9781107002173.

[20] S. M. Barnett. *Quantum Information*. Oxford University Press Inc., 2009. URL: global.oup.com/academic/product/quantum-information-9780198527626.

[21] M. L. Bellac. *A Short Introduction to Quantum Information and Quantum Computation*. 1st. Cambridge University Press, 2006. URL: cambridge.org/9780521860567.

[22] A. Pathak. *Elements of Quantum Computation and Quantum Communication*. CRC Press, 2013. URL: crcpress.com/9781466517912.

[23] E. E. Rosinger. *Basics of Quantum Computation*. 2004. URL: arxiv.org/abs/quant-ph/0407064.

[24] J. A. Jones e D. Jaksch. *Quantum Information, Computation and Communication*. Cambridge University Press, 2012. URL: cambridge.org/9781107014466.

[25] A. Ekert, P. Hayden e H. Inamori. «Basic concepts in quantum computation». Em: United Kingdom, 2000. URL: link.springer.com/chapter/10.1007/3-540-45338-5_10.

[26] A. Y. Kitaev, A. H. Shen e M. N. Vyalyi. *Classical and Quantum Computation*. American Mathematical Society, 2002. ISBN: ISBN: 978-0-8218-3229-5.

[27] A. Barenco, A. Ekert, K.-A. Suominen e P. Törmä. «Approximate Quantum Fourier Transform and Decoherence». Em: *Phys. Rev. A* 54 (1996), pp. 139–146. DOI: 10.1103/PhysRevA.54.139. URL: https://link.aps.org/doi/10.1103/PhysRevA.54.139.

[28] T. G. Draper. «Addition on a Quantum Computer». Em: (1998). arXiv: 0008033v1 [quant-ph].

[29] L. Ruiz-Perez e J. C. Garcia-Escartin. «Quantum arithmetic with the Quantum Fourier Transform». Em: *Quantum Information Processing* 16 (2017). DOI: 10.1007/s11128-017-1603-1. URL: https://doi.org/10.1007/s11128-017-1603-1.

[30] S. Beauregard. «Circuit for Shor's algorithm using 2n+3 qubits». Em: *Quantum Information and Computation* 3 (2003), pp. 175–185. ISSN: 1533-7146.

[31] A. G. Fowler, S. J. Devitt e L. C. L. Hollenberg. «Implementation of Shor's Algorithm on a Linear Nearest Neighbour Qubit Array». Em: *Quantum Information and Computation* (2004), pp. 237–251. ISSN: 1533-7146.

[32] P. W. Shor. «Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer». Em: *SIAM Journal on Scientific and Statistical Computing* 26 (1997), pp. 1484–1509. DOI: 10.1137/S0097539795293172. URL: https://doi.org/10.1137/S0097539795293172.

[33] T. Takagi, ed. *Post-Quantum Cryptography:* 1ª ed. Fukuoka: Springer International Publishing, 2016. DOI: 10.1007/978-3-319-29360-8. URL: springer.com/gp/book/9783319293592.

[34] S. B. Wicker e V. K. Bhargava, eds. *Reed-Solomon Codes and Their Applications*. Wiley-IEEE Press, 1999. ISBN: 978-0-7803-5391-6. URL: www.wiley.com/WileyCDA/WileyTitle/productCd-0780353919,miniSiteCd-IEEE2.html.

[35] P. Almeida e D. Napp. «A new class of convolutional codes and its use in the McEliece Cryptosystem». Em: *Submited to Finite Fields and Applications* (2018). arXiv: 1804.08955v1.

[36] C. Peters. «Information-set decoding for linear codes over Fq». Em: *Post-Quantum Cryptography. PQCrypto 2010. Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg, 2010, pp. 81–94. ISBN: 978-3-642-12928-5.

[37] M. Baldi, M. Bianch, F. Chiaraluce, J. Rosenthal e D. Schipani. «Enhanced Public Key Security for the McEliece Cryptosystem». Em: *CRYPTOLOGY* (2012). DOI: 10.1007/s00145-014-9187-8.

[38] R. W. Hamming. *Coding and Information Theory*. Prentice Hall, 1980. ISBN: 9780131391390. URL: https://www.abebooks.com/9006638603/bd.