Wallace Mielniczki

Mecanismo de Criptografia Pós-Quântica ML-KEM: Uma Introdução para Estudantes de Ciência da Computação

Joinville

2023

UNIVERSIDADE DO ESTADO DE SANTA CATARINA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Wallace Mielniczki

MECANISMO DE CRIPTOGRAFIA PÓS-QUÂNTICA ML-KEM: UMA INTRODUÇÃO PARA ESTUDANTES DE CIÊNCIA DA COMPUTAÇÃO

Trabalho de conclusão de curso submetido à Universidade do Estado de Santa Catarina como parte dos requisitos para a obtenção do grau de Bacharel em Ciência da Computação

Karina Girardi Roggia
Orientadora

Joinville, Dezembro de 2023

MECANISMO DE CRIPTOGRAFIA PÓS-QUÂNTICA ML-KEM: UMA INTRODUÇÃO PARA ESTUDANTES DE CIÊNCIA DA COMPUTAÇÃO

Wallace Mielniczki

Trabalho de Conclusão de Curso apresentada ao curso de Bacharelado em Ciência da Computação do Centro de Ciências Tecnológicas da Universidade do Estado de Santa Catarina, como requisito parcial para a obtenção do grau de Bacharel em Ciência da computação.

Banca Examinadora

Karina Girardi Roggia - Doutora (orientadora)

Debora Cabral Nazário - Doutora

Charles Christian Miers - Doutor

Rafael Rodrigues Obelheiro - Doutor

Resumo

Em 1994, Peter Shor apresentou um algoritmo que resolve o problema da fatoração de números inteiros e do logaritmo discreto em tempo polinomial por um computador quântico, tais problemas são a base da segurança de algoritmos de criptografia assimétrica amplamente utilizados, tais como RSA e ECDH. Com o avanço nas pesquisas na área da computação quântica, iniciou-se uma preocupação com relação à segurança dos dados que dependem de sistemas criptográficos comprometidos pelo algoritmo de Shor. Em 2016, o *National Institute of Standards and Technology* (NIST) iniciou o programa *Post-Quantum Cryptography Standardization*, cujo objetivo é selecionar algoritmos de criptografia assimétricos e de assinatura digital resistentes a ataques realizados por computadores clássicos e quânticos, que possam ser utilizados em computadores convencionais. Dentre os algoritmos selecionados para padronização em 2022, está o algoritmo *Module-Lattice-based Key-Encapsulation Mechanism* (ML-KEM) para criptografia assimétrica, que é abordado nesse trabalho. Este trabalho visa servir como um material de estudos para acadêmicos de ciência da computação sobre o algoritmo ML-KEM e sua relação com reticulados.

Palavras-chave: criptografia, criptografia assimétrica, criptografia pós-quântica, reticulados.

Abstract

In 1994 Peter Shor presented an algorithm that solves the factorization problem of integers and the discrete logarithm in polynomial time by a quantum computer, such problems are the basis of the security of widely used asymmetric cryptography algorithms, such as RSA and ECDH. Due to quantum computing research advances, a concern has arisen regarding the security of data that depend on cryptographic systems compromised by Shor's algorithm. In 2016, the National Institute of Standards and Technology (NIST) started the Post-Quantum Cryptography Standardization program, whose objective is to select asymmetric cryptography and digital signature algorithms resistant to attacks performed by classical and quantum computers, which can be used in conventional computers. Among the algorithms selected for standardization in 2022 is the Module-Lattice-based Key-Encapsulation Mechanism (ML-KEM) algorithm for asymmetric cryptography that will be addressed in this work. This work aims to serve as a study material on the ML-KEM algorithm and its relation to lattices, for computer science academics.

Keywords: cryptography, asymmetric cryptography, post-quantum cryptography, lattices.

Lista de Figuras

Lista de Abreviaturas

CRYSTALS *Cryptographic Suite for Algebraic Lattices*

CVP *Closest Vector Problem*

DSA *Digital Signature Algorithm*

ECC *Elliptic-curve cryptography*

ECDH *Elliptic-curve Diffie–Hellman*

ECDSA *Elliptic Curve Digital Signature Algorithm*

GapSVP *Gap Shortest Vector Problem*

GNFS *General Number Field Sieve*

KEM *Key-Encapsulation Mechanism*

LWE *Learning With Errors*

ML-KEM *Module-Lattice-based Key-Encapsulation Mechanism*

MLWE *Module Learning With Errors*

NIST *National Institute of Standards and Technology*

NP-HARD *Non-deterministic Polynomial-time Hardness*

NTT *Number Theoretic Transform*

PQC *Post-Quantum Cryptography*

RLWE *Ring Learning With Errors*

RSA *Rivest-Shamir-Adleman*

SIVP *Shortest Independent Vector Problem*

SVP *Shortest Vector Problem*

Lista de Tabelas

Sumário

1 Introdução

Em 1994 Peter Shor apresentou um algoritmo que resolve o problema da fatoração de números inteiros e do logaritmo discreto em tempo polinomial por um computador quântico (SHOR, 1994), tais problemas são a base da segurança de algoritmos de criptografia assimétrica amplamente utilizados, como *Rivest-Shamir-Adleman* (RSA) e *Elliptic-curve Diffie–Hellman* (ECDH). Algumas empresas como IBM, Microsoft, Intel e Google estão investindo fortemente na pesquisa de computadores quânticos e tecnologias relacionadas. Em 2019 o Google anunciou a conquista da supremacia quântica, quando um computador quântico supera em alguma tarefa um computador clássico, resolvendo um problema em 200 segundos em que um supercomputador clássico resolveria em aproximadamente 10000 anos (ARUTE, 2019). Uma equipe de pesquisadores da China fatorou uma chave de 48 bits usando um computador quântico de 10 qubit, a equipe estimou fatorar uma chave de 2048 bits com um computador de 372 qubits usando seu algoritmo(YAN et al., 2022), entretanto a IBM já possui um computador quântico com um processador de 433 qubit (GAMBETTA, 2022).

Com o avanço nas pesquisas na área da computação quântica, iniciou-se uma preocupação com relação à segurança dos dados que dependem de sistemas criptográficos comprometidos pelo algoritmo de Shor. Em 2016 o NIST iniciou o programa *Post-Quantum Cryptography Standardization*, cujo objetivo é selecionar algoritmos de criptografia assimétricos e de assinatura digital resistentes a ataques realizados por computadores clássicos e quânticos, que possam ser utilizados em computadores convencionais (NIST, 2017). Os algoritmos participantes são estudados e avaliados pela comunidade científica, considerando a segurança, eficiência e interoperabilidade com outros protocolos de criptografia atuais. O processo de padronização é realizado em rodadas, os algoritmos mais promissores são selecionados para prosseguir para a próxima rodada.

Na primeira rodada foram submetidos 45 esquemas de criptografia assimétrica e 19 de assinatura digital. A Tabela 1.1 mostra os diferentes tipos de criptografia assimétrica e de assinatura digital submetidos nessa rodada do programa, dos quais após 3 rodadas foram selecionados para padronização e substituição em larga escala os algoritmos mostrados na Tabela 1.2. Outros algoritmos prosseguiram para a quarta rodada,

na qual continuam sendo estudados e avaliados para uma possível padronização futuramente. Embora os algoritmos baseados em reticulados apresentem uma ótima segurança e desempenho, não se é recomendado depender apenas de um tipo de criptografia, este é o motivo da padronização do algoritmo SPHINCS+ e do estudo dos algoritmos da quarta rodada, conforme pode ser observado na Tabela 1.3.

Tabela 1.1: Primeira rodada do NIST PQC.

Tipo	Criptografia assimétrica	Assinatura digital
Baseada em reticulados	21	5
Baseada em código	17	2
Baseada em *hash*	0	3
Multivariada	2	7
Outros	5	2

Fonte: Adaptado de: (MOODY, 2019).

Tabela 1.2: Algoritmos selecionados do NIST PQC.

Tipo	Criptografia assimétrica	Assinatura digital
Baseada em reticulados	CRYSTALS-Kyber	CRYSTALS-Dilithium FALCON
Baseada em *hash*		SPHINCS+

Fonte: Adaptado de: (MOODY, 2019).

Tabela 1.3: Quarta rodada do NIST PQC.

Tipo	Criptografia assimétrica
Baseada em código	BIKE Classic MacEliece HQC
Baseada em isogenia	SIKE

Fonte: Adaptado de: (MOODY, 2019).

A equipe *Cryptographic Suite for Algebraic Lattices* (CRYSTALS) desenvolveu dois sistemas criptográficos, escolhidos para padronização após avaliações da comunidade científica no programa *Post-Quantum Cryptography Standardization* do NIST, sendo o algoritmo Kyber para criptografia de chave assimétrica e Dilithium para assinatura digital (ALAGIC; APON; COOPER, 2022). Após a aprovação e padronização do Kyber no

programa PQC em 24 de agosto de 2023, o NIST alterou o nome do algoritmo Kyber para ML-KEM. O algoritmo ML-KEM, que é o foco deste trabalho, é baseado no esquema LWE introduzido por (REGEV, 2009), tal modelo ficou famoso devido a sua relação com alguns problemas de difícil resolução baseados em reticulados, como *Shortest Vector Problem* (SVP) e *Closest Vector Problem* (CVP), que possuem grande complexidade de resolução mesmo para um computador quântico.

Devido ao fato da popularidade da criptografia baseada em reticulados ser recente, existe uma escassez de materiais didáticos acessíveis destinados a estudantes de ciência da computação interessados no algoritmo ML-KEM e nos conceitos matemáticos associados. O propósito deste trabalho é fornecer um material de estudo de fácil compreensão que aborda o algoritmo de criptografia pós-quântico ML-KEM e os princípios matemáticos subjacentes, tornando-os acessíveis a estudantes de computação. Os objetivos específicos deste trabalho incluem:

- Apresentar uma introdução à criptografia, contextualizando seus diversos tipos;
- Explorar os conceitos de reticulados e suas aplicações na criptografia, fornecendo uma base sólida para compreensão;
- Especificar e compreender os principais problemas computacionais envolvendo reticulados;
- Oferecer todos os conceitos para o entendimento do algoritmo ML-KEM de forma simplificada.
- Realizar uma implementação simplificada do algoritmo ML-KEM.

A implementação simplificada na linguagem C do algoritmo ML-KEM pode ser encontrada no repositório do GitHub em https://github.com/opallace/ML-KEM.

O desenvolvimento deste trabalho se deu por meio de uma pesquisa referenciada e aplicada, com o uso de diversas pesquisas pré-existentes como base teórica para o desenvolvimento do texto apresentado neste documento.

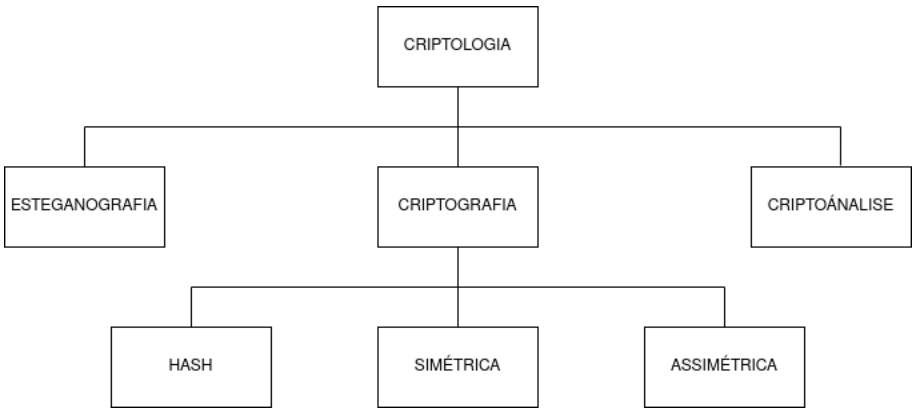
A estrutura do texto consiste em cinco capítulos, no qual o primeiro capítulo introduz o problema, motivação e os objetivos deste trabalho. O Capítulo 2 apresenta alguns conceitos básicos de criptografia e criptoanálise, fundamentais para o entendimento

da área. No Capítulo 3 é abordada a estrutura algébrica de reticulados, que possui problemas computacionais que compõem a base da segurança do algoritmo ML-KEM. A explicação do algoritmo ML-KEM está no Capítulo 4, sendo que, na Seção 4.1 é explicado o algoritmo K-PKE, utilizado como um subprocesso do ML-KEM. Por fim, seguem-se as conclusões e trabalhos futuros abertos a partir deste no Capítulo 5. Podem ser encontrados, nos apêndices, os conceitos, demonstrações matemáticas e os algoritmos abordados.

2 Criptografia

A criptologia é a ciência das comunicações secretas (DOOLEY, 2018) que contempla diversas técnicas para garantir uma comunicação segura entre um emissor e seu destinatário, através de um canal de comunicação suscetível a interceptações por terceiros não autorizados. A criptologia possui três subáreas: esteganografia, criptografia e criptoanálise. A esteganografia visa ocultar fisicamente uma mensagem, como em áudios, imagens ou papel, sem alterar sua apresentação. A criptografia, por sua vez, altera a apresentação da mensagem, tornando-a ilegível para quem não souber reverter o processo. A criptoanálise é o estudo da segurança dos métodos de criptografia. A Figura 2.1 ilustra esta divisão de áreas.

Figura 2.1: Representação parcial das áreas da criptologia.



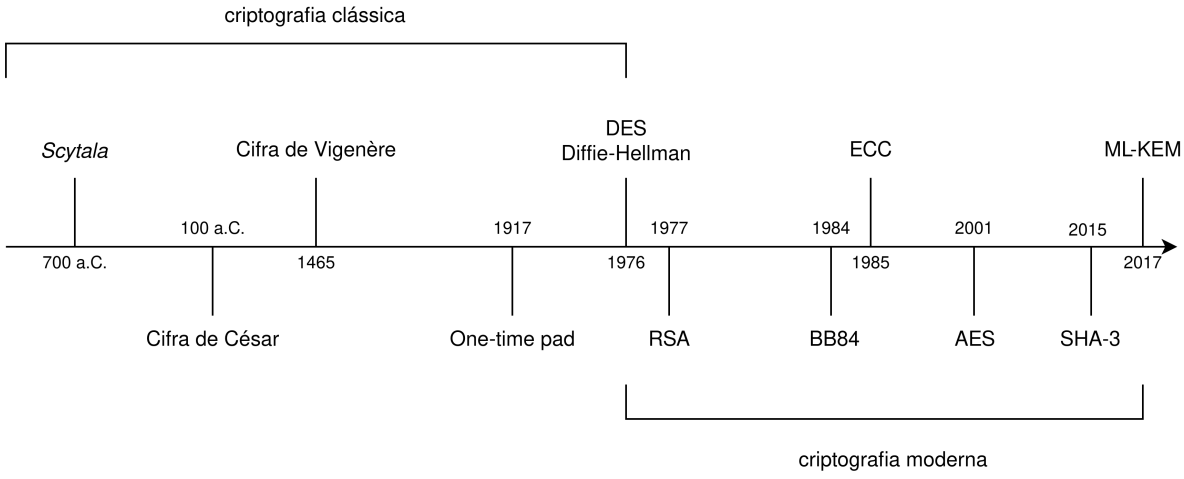
Fonte: O autor.

A criptografia possui os seguintes objetivos: confidencialidade, integridade, autenticação e irretratabilidade (DELFS; KNEBL, 2007). A confidencialidade visa garantir que um determinado dado possa ser lido apenas por destinatários autorizados. Integridade procura assegurar que um dado não seja modificado durante a transmissão. A autenticação determina que um destinatário possa verificar a origem da mensagem. Por fim, a irretratabilidade certifica que um emissor não possa negar a autoria de uma mensagem. Por si só, um algoritmo de criptografia não assegura estes objetivos e outras necessidades de segurança. Dependendo do cenário, é necessário utilizar diversos algoritmos em conjunto, cada um resolvendo uma parcela do problema. Por exemplo, ao enviar uma mensagem utilizando o ML-KEM, iremos garantir apenas confidencialidade e inte-

gridade. Para garantir a autenticação e irretratabilidade devemos utilizar um algoritmo de assinatura digital.

Historicamente a criptografia é dividida em dois períodos: criptografia clássica e moderna. Alguns autores como (STALLINGS, 2008; KATZ; LINDELL, 2021) consideram a criptografia clássica os métodos de criptografia que antecedem a criação da criptografia assimétrica em 1976 por Whitfield Diffie e Martin Hellman, isto é, métodos de transposição, substituição, *one-time pad*, máquinas de rotor, etc. A Figura 2.2 apresenta alguns dos sistemas de criptografia clássicos e modernos. Entre os anos de 1970 e 1980, a criptografia teve um significativo avanço com a convergência da matemática e computação na área (KATZ; LINDELL, 2021), se destacando a publicação do DES (STANDARDS, 1977) e troca de chaves Diffie-Hellman (DIFFIE; HELLMAN, 1976). A partir desse período, subdisciplinas da matemática, como álgebra abstrata, teoria dos números e matemática finita; e da computação, como teoria da informação e complexidade computacional, passaram a estar presentes na criação e na utilização dos processos de criptografia, assim dando início à criptografia moderna.

Figura 2.2: Linha do tempo da criptografia.



Fonte: O autor.

É importante destacar a diferença entre criptografia quântica e pós-quântica, embora os termos sejam semelhantes e podem causar uma certa confusão, eles se diferem em seu propósito e na natureza de como realizam os processos de criptografia. Os algoritmos quânticos como o BB84, desenvolvido por Charles Bennett e Gilles Brassard (BENNETT; BRASSARD, 2014), utilizam as propriedades da mecânica quântica como método de criptografia. Ou seja, tem como objetivo serem executados em computadores

quânticos para garantirem a segurança dos dados. A criptografia pós-quântica é um termo que surgiu devido a preocupação com a segurança das informações criptografadas com algoritmos baseados em problemas que podem ser resolvidos pelo algoritmo de Shor (SHOR, 1994). Estes algoritmos de criptografia pós-quânticos tem como objetivo serem executados em computadores convencionais, e serem seguros tanto contra ataques realizados por computadores clássicos quanto quânticos (NIST, 2017), como é o caso do ML-KEM.

A criptografia possui três subáreas principais, as funções *hash*, criptografia simétrica e assimétrica. É importante conhecer as particularidades de cada uma dessas áreas, como, seu funcionamento, s problemas e soluções que os algoritmos destas áreas resolvem. Cada uma delas possui suas vantagens e desvantagens, com isso, são aplicáveis a problemas específicos na criptografia.

2.1 Função Hash

Uma função *hash* é uma função que mapeia uma entrada de tamanho variável em uma saída de tamanho fixo, $f : \{0,1\}^* \rightarrow \{0,1\}^n$. Entretanto, essas funções devem possuir algumas características para serem consideradas funções *hash*:

(i) saída de tamanho fixo: independente do tamanho da entrada, a saída deve sempre ter o mesmo tamanho;

(ii) determinístico: para uma mesma entrada, deve-se sempre obter a mesma saída;

(iii) eficiência de operação: a operação de mapeamento deve ser rápida.

Funções *hash* não criptográficas são comumente utilizadas onde a segurança dos dados não é uma necessidade, por exemplo, na verificação e integridade de dados, tabelas *hash*, etc. Por sua vez, as funções *hash* criptográficas devem possuir propriedades adicionais para terem aplicabilidade na criptografia, essas propriedades são:

(i) resistência à pré-imagem: deve ser inviável reverter o processo de *hash*, ou seja, descobrir uma entrada correspondente ao *hash*;

(ii) resistência à segunda pré-imagem: dado uma entrada e seu *hash*, deve ser difícil achar outra entrada com mesmo *hash*;

(iii) resistência à colisão: deve ser difícil encontrar duas entradas escolhidas arbitrariamente que possuam o mesmo *hash*.

A complexidade de se resolver o problema do item (ii) no pior caso é de $\mathcal{O}(2^n)$, isto porque seria necessário verificar todos os resultados possíveis da função *hash*, até encontrar um resultado compatível com o *hash* de entrada. Já a complexidade de se resolver o problema do item (iii) é de $\mathcal{O}(2^{n/2})$, uma vez que a probabilidade de se encontrar duas mensagens diferentes escolhidas aleatoriamente que produzem o mesmo *hash* é maior que 50% em $2^{n/2}$ tentativas como demonstrado em (YUVAL, 1979; STALLINGS, 2008).

As funções *hash* criptográficas que possuem as propriedades citadas acima podem ser utilizadas na área da criptografia como, por exemplo, em armazenamento de senhas. Perceba que se fosse aplicada uma função *hash* não criptográfica qualquer a uma senha e armazenada em um banco de dados, a mesma poderia ser decifrada, já que seria possível reverter o processo através da não resistência à pré-imagem. Outras aplicabilidades de funções *hash* são em geradores de números pseudoaleatórios e como subprocessos em outros algoritmos de criptografia.

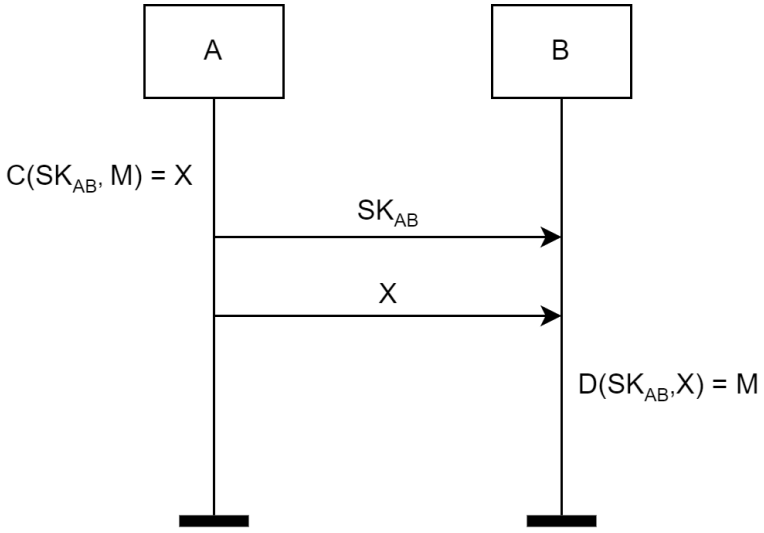
2.2 Criptografia simétrica

A criptografia simétrica tem como principal característica a utilização de uma mesma chave criptográfica para a realização dos processos de cifragem e decifragem. Diferente das funções *hash* criptográficas, em que o processo de decifragem não deve ser possível computacionalmente, um algoritmo de criptografia simétrico é projetado para realizar a cifragem e decifragem de uma mensagem usando uma mesma chave criptográfica. Uma mensagem cifrada utilizando um algoritmo simétrico somente pode ser decifrada com a mesma chave que a cifrou. Alguns dos algoritmos de criptografia simétrica mais utilizado é o AES (DWORKIN et al., 2001).

A Figura 2.3 ilustra um exemplo da utilização de um algoritmo de cifragem simétrica para o envio de uma mensagem M, no qual C é o algoritmo de cifragem, D é o algoritmo de decifragem, SK_{AB} é a chave simétrica compartilhada entre A e B, e X é a mensagem cifrada.

Existem dois tipos de cifras simétricas quanto à forma que uma mensagem é processada: cifras de bloco e cifras em fluxo. As cifras de bloco dividem a mensagem de

Figura 2.3: Envio de mensagem usando criptografia simétrica.



Fonte: O autor.

entrada em blocos de tamanho fixo e processam individualmente cada uma destas, para cada bloco processado se tem um bloco resultante. Caso o tamanho da mensagem não seja divisível pelo tamanho do bloco, é necessário inserir um preenchimento na mensagem de entrada. A cifra AES, por exemplo, pode operar em blocos de 128, 192 ou 256 bits. A Figura 2.4 ilustra uma cifra de bloco de 128 bits.

As cifras em fluxo, por sua vez, processam os elementos da entrada continuamente, geralmente bit a bit, ou bytes a byte, e produzem uma saída conforme processam esses elementos. A operação $\oplus$ (XOR/ou exclusivo) é um exemplo de cifra de fluxo, na qual a cifragem consiste na aplicação da operação $\oplus$ entre um byte/mensagem e uma chave secreta de um byte,

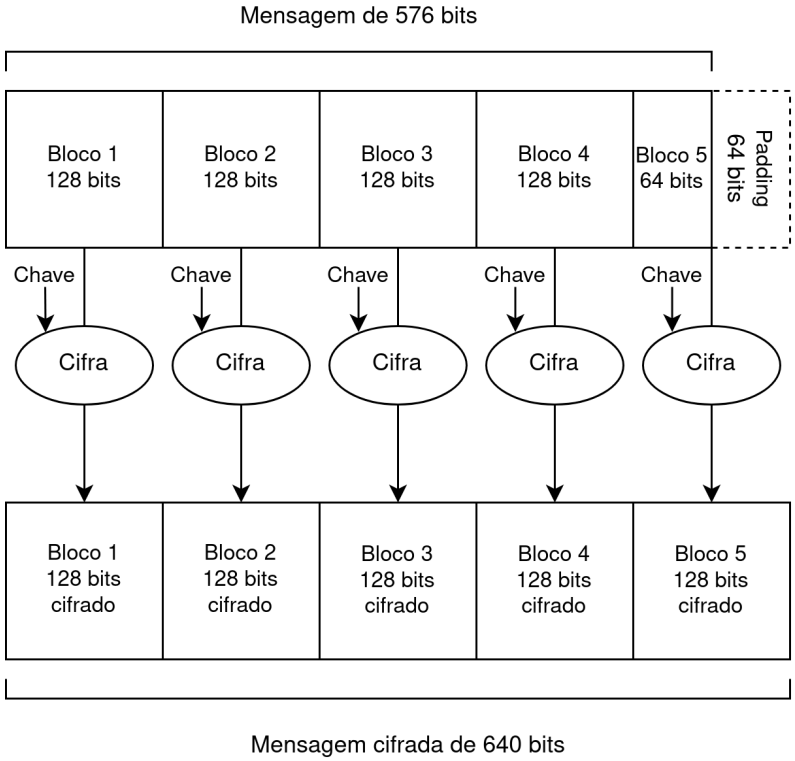
$$\begin{array}{cll} & 10010110 & \text{mensagem} \\ \oplus & 01101100 & \text{chave secreta} \\ \hline & 11111010 & \text{mensagem cifrada} \end{array}$$

e a decifragem é realizada simplesmente aplicado novamente a operação $\oplus$ entre o byte cifrado e a mesma chave secreta.

$$\begin{array}{cll} & 11111010 & \text{mensagem cifrada} \\ \oplus & 01101100 & \text{chave secreta} \\ \hline & 10010110 & \text{mensagem} \end{array}$$

A principal limitação das cifras simétricas é que, para transmitir uma mensagem cifrada, é necessário também transmitir para a outra parte a chave que decifra

Figura 2.4: Cifra de bloco.



Fonte: O autor.

esta mensagem, como pode ser visto na Figura 2.3. O fato de ter que enviar a chave por um meio de comunicação não seguro, torna necessária a utilização em conjunto com um algoritmo de cifragem assimétrica, onde este tipo de algoritmo possui a particularidade de possuir chaves diferentes para cifrar e decifrar.

A segurança dos algoritmos de criptografia simétricos não sofreram impacto significativo pela computação quântica. Atualmente, apenas o algoritmo de Grover (GROVER, 1996) reduz a complexidade computacional dos problemas utilizados pelos algoritmos de cifragem simétrica. Em 1996, Lov Grover publicou um algoritmo quântico capaz de realizar a busca de um valor em uma lista de n elementos não ordenados com complexidade $\mathcal{O}(\sqrt{n})$, enquanto computadores clássicos conseguem realizar esta busca em $\mathcal{O}(n)$. Isto significa que um computador quântico pode encontrar uma chave simétrica de 128 bits realizando 2^{64} operações com o algoritmo de Grover, ao invés de 2^{128} por um computador clássico. Note que para manter a mesma segurança de uma chave simétrica de 128 bits, devemos alterar o tamanho da chave para 256 bits, desta forma um computador quântico terá que realizar 2^{128} operações, a mesma quantidade que se tinha com a computação clássica.

2.3 Criptografia assimétrica

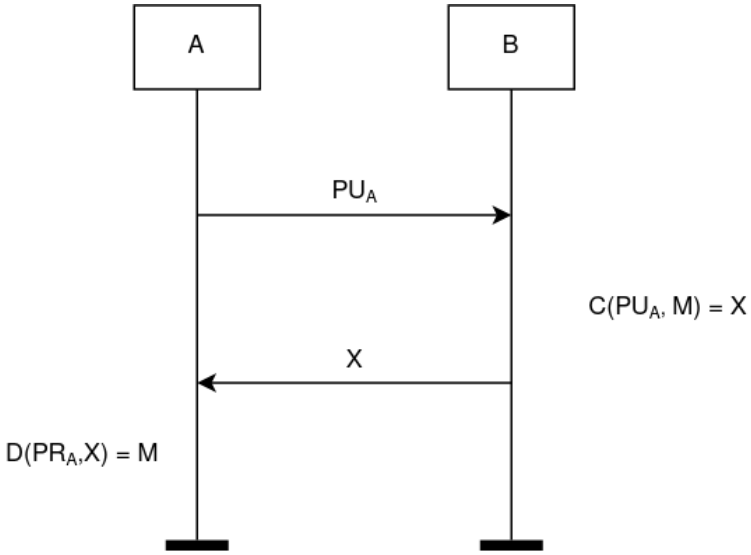
A criptografia assimétrica, também conhecida por criptografia de chave pública, foi inspirada pelo algoritmo de troca de chaves Diffie-Hellman em 1976. Tradicionalmente, para duas partes possuírem a mesma chave simétrica, era necessário que uma delas enviasse sua chave para a outra parte, entretanto, na maioria dos sistemas, não se tem a garantia de um canal de comunicação seguro para essa troca. O algoritmo de Diffie-Hellman (DIFFIE; HELLMAN, 1976) foi proposto como uma solução para este problema, estabelecendo uma chave simétrica compartilhada entre duas partes sem a necessidade do envio desta por um meio de comunicação não seguro. O algoritmo de Diffie-Hellman, diferente dos algoritmos simétricos, utiliza dados privados que não devem ser compartilhados, e dados públicos que podem ser compartilhados. No final do processo, ambas as partes possuem uma informação em comum (chave simétrica), que é muito custosa de ser obtida computacionalmente a partir das informações públicas transmitidas pelo meio de comunicação.

Em 1977, Ron Rivest, Adi Shamir e Leonard Adleman publicaram o primeiro algoritmo de criptografia assimétrica, chamado RSA (RIVEST; SHAMIR; ADLEMAN, 1978). Diferente do algoritmo de troca de chaves Diffie-Hellman, o RSA realiza a cifragem de mensagens, e não apenas o estabelecimento de uma chave simétrica em comum. Em um algoritmo de criptografia assimétrica como o RSA, a cifragem de uma mensagem somente pode ser realizada por uma chave pública, e a decifragem apenas pela chave privada. Desse modo, o envio de mensagens por um canal de comunicação pode ser realizado de forma segura, já que a chave privada não necessita ser transmitida para outra parte. A Figura 2.5 exemplifica esse processo, supondo que A tenha gerado suas chaves públicas e privadas previamente, A envia sua chave pública para B, que por sua vez cifra uma mensagem usando a chave pública de A e envia para A esta mensagem cifrada, A decifra a mensagem com sua chave privada. Assim como o algoritmo Diffie-Hellman, é custoso obter computacionalmente os dados privados a partir dos dados públicos nos algoritmos de criptografia assimétricos.

2.4 Encapsulamento de chave

Os algoritmos de criptografia assimétricos são projetados para cifrar mensagens de tamanho limitado, enquanto algoritmos simétricos são projetados para cifrar mensagens de

Figura 2.5: Envio de mensagem usando criptografia assimétrica.

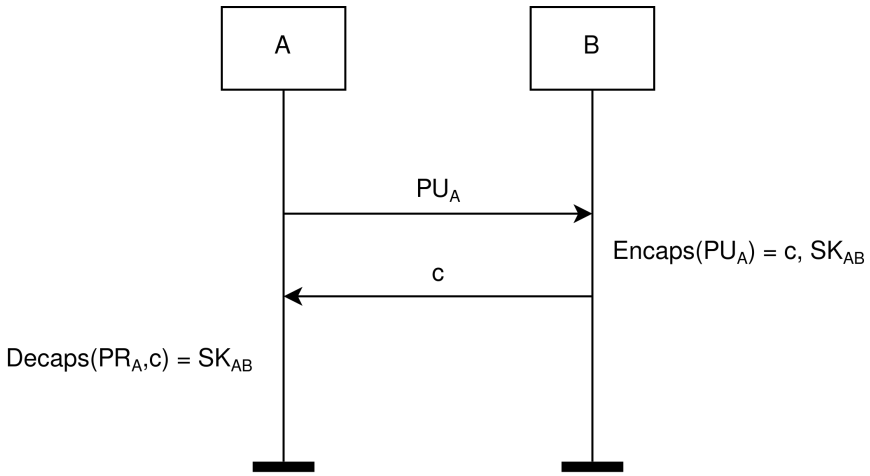


Fonte: O autor.

tamanho ilimitado. Desta forma, para ser possível a comunicação de longas mensagens entre duas partes utilizando cifra simétrica, é necessária que ambas as partes tenham uma chave simétrica em comum. Atualmente, é utilizado o algoritmo de Diffie-Hellman sob a estrutura algébrica de curvas elípticas ECDH para estabelecer esta chave simétrica em comum, entretanto, este algoritmo não é considerado seguro contra computadores quânticos por se basear no problema do logaritmo discreto. Desta forma, é utilizada outra técnica para estabelecer uma chave simétrica comum entre duas partes, chamada *Key-Encapsulation Mechanism* (KEM). A principal diferença entre a troca de chaves Diffie-Hellman e KEM é que o algoritmo de Diffie-Hellman não permite a escolha de uma chave privada previamente, esta chave em comum é gerada ao longo do processo. O KEM, por sua vez, funciona de uma maneira diferente, uma das partes gera a chave simétrica e a cifra usando a chave pública da outra parte. Desta forma, ambas as partes possuem uma chave simétrica em comum para utilizar em um algoritmo simétrico, e este compartilhamento de chaves é estabelecido por um algoritmo de criptografia assimétrico, neste caso, o algoritmo assimétrico deve ser resistente à computação quântica, como o ML-KEM.

O funcionamento do KEM pode ser visto na Figura 2.6, na qual A envia sua chave pública PU_A para B, B gera a chave simétrica comum SK_{AB} e a cifra usando a chave pública de A, gerando c que é enviado para A. A, por sua vez, decifra c usando sua chave privada PR_A, gerando SK_{AB}.

Figura 2.6: Encapsulamento de uma chave privada em comum.



Fonte: O autor.

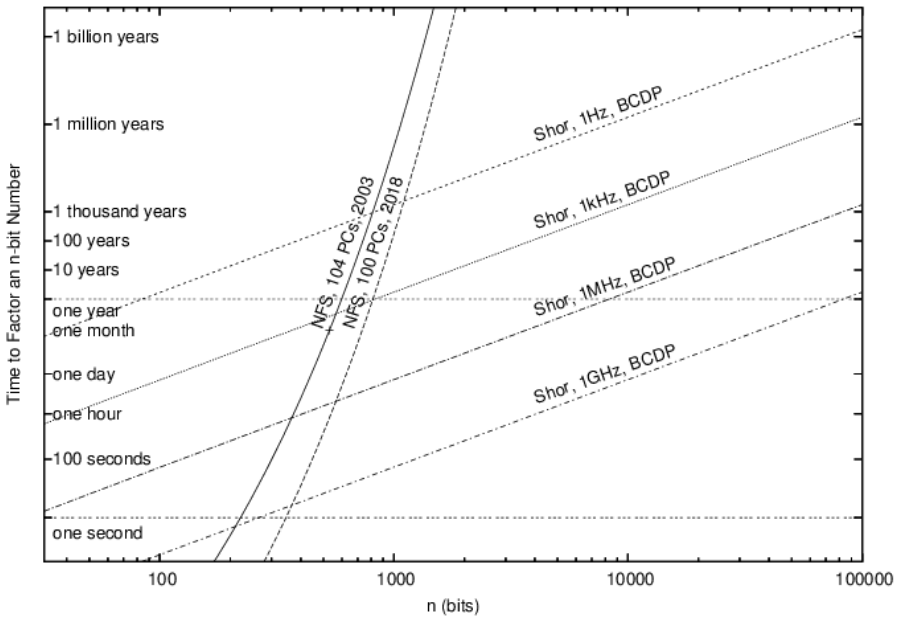
2.5 Problemas computacionais

Alguns algoritmos de criptografia assimétrica, como Diffie-Hellman e ECDH (Troca de chaves), RSA e *Elliptic-curve cryptography* (ECC) (Cifragem) e RSA, *Elliptic Curve Digital Signature Algorithm* (ECDSA) e *Digital Signature Algorithm* (DSA) (Assinatura digital) possuem sua segurança baseadas nos problemas da fatoração de números inteiros e do logaritmo discreto.

Eric Bach em (CALIFORNIA; BACH, 1984) demonstra uma redução probabilística em tempo polinomial do problema da fatoração prima para o problema do logaritmo discreto e do problema do logaritmo discreto para o problema da fatoração prima. Isso significa que se a fatoração prima de um número n for realizada, então existe uma alta probabilidade de se resolver o problema do logaritmo discreto com módulo n e vice-versa. Isso implica que a complexidade de resolver ambos os problemas algoritmicamente é a mesma, se for desconsiderado o tempo da redução.

A complexidade de se resolver os problemas da fatoração prima e do logaritmo discreto em um computador clássico é $\mathcal{O}(e^{(log\ n)^{1/3}(log\ (log\ n))^{2/3}})$ com o algoritmo GNFS, considerado o algoritmo clássico mais eficiente na resolução desses problemas, enquanto o algoritmo de Shor possui a complexidade de $\mathcal{O}((log\ n)^2(log\ log\ n)(log\ log\ log\ n))$(HAMDI et al., 2014). A Figura 2.7 mostra um gráfico comparativo de tempo de execução entre o algoritmo de Shor e GNFS.

Figura 2.7: Comparativo entre os algoritmos Shor e GNFS na resolução do problema da fatoração prima de um número composto.



Fonte: (HAMDI et al., 2014).

2.5.1 Problema da fatoração prima

O problema da fatoração prima de inteiros consiste em, dado um número composto, encontrar seus fatores primos. Tem-se, como exemplo, de uso de propriedades deste problema, a geração de chaves no sistema de criptografia RSA como pode ser visto no Algoritmo 1. A chave pública n é um número composto gerado pelo produto de dois números primos; fatorando o número n, descobrem-se os números primos p e q. Obtendo os números p e q, pode-se derivar a chave privada completa, visto que o cálculo da chave privada d necessita de e, que é uma chave pública, e $\Phi(n)$, que é obtido através de p e q.

Algoritmo 1: Geração de chaves RSA.

Saída: Chave pública: (n, e) e Chave privada: (p, q, d)

1: Escolha dois números primos p e q

2: Compute $n = p.q$

3: Compute $\Phi(n) = (p-1)(q-1)$

4: Escolha um expoente e tal que $1 \leq e < \Phi(n)$ e $mdc(e, \Phi(n)) = 1$

5: Compute d tal que $d.e \equiv 1$ (**mod** $\Phi(n)$)

Este é um exemplo de como a existência de um algoritmo que fatora um número composto em tempo polinomial afeta a segurança de um dos principais algoritmos de chave

pública utilizados.

2.5.2 Problema do logaritmo discreto

O problema do logaritmo discreto consiste em, dado um número primo p, sua raiz primitiva r e um número n tal que $0 \leq n < p$, encontrar o expoente x que satisfaça a equação modular $n \equiv r^x \pmod{p}$. Os possíveis valores que satisfazem a equação $n \equiv r^x \pmod{p}$ compõem o conjunto finito $\mathbb{Z}_p = \{0, 1, 2, ..., p-1\}$, e a utilização de uma raiz primitiva de p como base faz com que os possíveis resultados sejam uniformemente distribuídos, isto é, de igual probabilidade de serem gerados. Isso faz com que, para encontrarmos o expoente que satisfaça a equação $n \equiv r^x \pmod{p}$, seja necessário se verificar todos os possíveis valores de x.

Supondo que um emissor deseja realizar troca de chaves usando o algoritmo de Diffie-Hellmann indicado pelo Algoritmo 2, resolvendo as equações $A \equiv g^a \pmod{p}$ e $B \equiv g^b \pmod{p}$ é possível derivar a chave secreta s.

Algoritmo 2: Troca de chaves Diffie-Hellman.

Saída: Chave pública: (p, g, A, B), Chave privada lado A: (a, s), Chave privada lado B: (b, s)

1: O lado A e B entram em acordo publicamente para utilizar os números primos p e g co-primos, tal que $p > g$.

2: O lado A escolhe um número a e computa $A = g^a \bmod p$

3: O lado B escolhe um número b e computa $B = g^b \bmod p$

4: Ambos os lados trocam entre si os valores A e B

5: O lado A computa $s = B^a \bmod p$

6: O lado B computa $s = A^b \bmod p$

Este é outro exemplo de como um algoritmo que resolve o problema do logaritmo discreto pode afetar um algoritmo de criptografia que baseia sua segurança na dificuldade de se resolver o problema do logaritmo discreto.

2.6 Criptoanálise

Quando um algoritmo de criptografia é projetado, é necessário um estudo deste algoritmo antes de ser implementado em cenários reais, para que se tenha confiança de sua segurança quando aplicado em dados sensíveis. A subárea da criptologia que visa estudar a segurança de algoritmos de criptografia é a criptoanálise. Devido à variedade de técnicas utilizadas e tipos de criptoanálise, esta seção será restrita apenas a princípios e conceitos básicos desta área, deixando as técnicas de criptoanálise referentes aos algoritmos relacionados a reticulados e ao ML-KEM em suas respectivas seções.

O criptógrafo holandês Auguste Kerckhoffs propôs seis princípios que deveriam ser levados em consideração ao se projetar um criptossistema militar para que este possa ser considerado seguro (KERCKHOFFS, 1883), são esses princípios:

1. O sistema deve ser materialmente, se não matematicamente, indecifrável;
2. É necessário que o sistema em si não requeira sigilo, e que não seja um problema se ele cair nas mãos do inimigo;
3. Deve ser possível comunicar e lembrar da chave sem a necessidade de notas escritas, e os interlocutores devem ser capazes de modificá-la a seu critério;
4. Deve ser aplicável à correspondência telegráfica;
5. O sistema deve ser portátil, e não deve exigir a participação de múltiplas pessoas na sua operação e manuseio;
6. Por fim, o sistema deverá ser simples de usar e não exigir conhecimentos profundos ou concentração dos seus usuários nem um conjunto complexo de regras.

Embora esses princípios foram criados para a tecnologia militar da época, eles ainda são válidos para os dias atuais com algumas modificações. Por exemplo, ao invés de dever ser aplicável à correspondência telegráfica, o criptossistema deve ser aplicável aos protocolos de comunicações atuais. O segundo item vai em oposição à segurança por obscuridade, cujo princípio é ocultar todo o funcionamento de um algoritmo, fornecendo o mínimo possível de informações sobre seu funcionamento; dessa forma, acredita-se que quanto menos informações for divulgada sobre o algoritmo de criptografia, mais seguro ele estará. Steven Bellovin e Randy Bush em (BELLOVIN; BUSH, 2002), argumentam que

algoritmos de criptografia que utilizam a obscuridade como técnica para garantir sua segurança podem ser considerados seguros a curto prazo, mas a longo prazo somente os algoritmos publicamente analisados podem ser considerados seguros. Segundo eles, ao se ocultar as vulnerabilidades de um algoritmo de criptografia, as probabilidades de que estas possam ser reparadas diminuem, e aumentam as probabilidades de que estas vulnerabilidades possam ser exploradas por um atacante mal-intencionado. Tornar público o funcionamento de um algoritmo de criptografia permite a análise deste algoritmo pela comunidade científica, assim expondo possíveis vulnerabilidades não descobertas, tornando os algoritmos de criptografia mais seguros e confiáveis.

Claude Shannon em (SHANNON, 1945) recomendou que um criptossistema deve possuir duas propriedades, confusão e difusão. O conceito de confusão em uma cifra significa que cada bit do texto cifrado deve depender de uma série de bits da chave criptográfica, dessa forma obscurecendo a relação entre a chave e o texto cifrado, o que dificulta recuperar informações da chave privada a partir do texto cifrado. A propriedade de difusão diz que, se alterado um único bit do texto original, o texto cifrado deve ser alterado em 50% ou mais, e essas alterações devem estar esparsas no texto cifrado. A ausência desta propriedade significaria que cada bit do texto original estaria relacionado unicamente com um bit do texto cifrado, facilitando um ataque por criptoanálise.

A Tabela 2.1 mostra alguns modelos de ataques a esquemas de criptografia com base nas informações disponíveis ao criptoanalista, para todos os casos é considerado que o atacante possui conhecimento do algoritmo de cifragem seguindo os princípios de Kerckhoffs. Esses modelos de ataques funcionam como uma espécie de jogo, em que são fornecidas informações ao atacante e, com base nessas informações, o atacante deve ser capaz de recuperar alguma informação útil do texto cifrado.

Os modelos de ataques são uma classificação muito utilizada para medir o nível de segurança de esquemas de criptografia. Por exemplo, se foi realizado um ataque malsucedido a um algoritmo de criptografia fornecendo apenas o texto cifrado ao atacante (COA), este algoritmo é considerado menos seguro comparado a outro algoritmo de criptografia que se teve um ataque malsucedido mesmo fornecendo mais informações como pares de textos originais e cifrados (KPA). Note que ambos os algoritmos passaram no teste de criptoanálise, entretanto o segundo algoritmo passou por uma análise mais rígida.

Tabela 2.1: Tipos de ataques a mensagens criptografadas.

Tipo de ataque	Conhecimento ao criptoanalista
Apenas texto cifrado (COA)	Texto cifrado(desafio)
Texto claro conhecido (KPA)	Texto cifrado(desafio) e uma ou mais pares de texto claro e texto cifrado por uma chave secreta
Texto claro escolhido (CPA)	Texto cifrado(desafio) e texto claro escolhido juntamente com o texto correspondente cifrado por uma chave secreta
Texto cifrado escolhido (CCA)	Texto cifrado(desafio) e textos cifrados escolhidos juntamente com os textos correspondentes decifrados por uma chave secreta
Texto cifrado escolhido adaptativamente (CCA2)	Texto cifrado(desafio) e textos cifrados escolhidos adaptativamente juntamente com os textos correspondentes decifrados por uma chave secreta
Texto escolhido	Texto cifrado, texto claro escolhido juntamente com o texto correspondente cifrado por uma chave secreta e texto cifrado pretendido, juntamente com seu texto claro correspondente decifrado por uma chave secreta

Fonte: Adaptado de (STALLINGS, 2008).

Outra classificação para esquemas de criptografia é a indistinguibilidade de textos cifrados, essa propriedade garante que, fornecidos pares de textos cifrados e decifrados, o atacante não deve conseguir identificar qual dos textos cifrados corresponde aos textos decifrados com probabilidade maior que 50% (KATZ; LINDELL, 2021). As três formas de indistinguibilidade de textos cifrados mais usadas na criptografia são:

- ***Indistinguishability under chosen plaintext attack* (IND-CPA):** O atacante envia um par de textos para um desafiador que cifra um destes textos escolhidos aleatoriamente, e envia este texto cifrado de volta para o atacante, que por sua vez deve deduzir com uma probabilidade maior que 50% qual dos textos foi cifrado.

- ***Indistinguishability under (non-adaptive) chosen ciphertext attack* (IND-CCA):** O atacante pode realizar cifragens e decifragens de textos escolhidos arbitrariamente por um oráculo, e envia um par de textos para um desafiador que cifra um destes textos escolhidos aleatoriamente, e envia este texto cifrado para o atacante, que por sua vez deve deduzir com uma probabilidade maior que 50% qual dos textos foi cifrado.

- ***Indistinguishability under adaptive chosen ciphertext attack* (IND-CCA2):** O atacante pode realizar cifragens e decifragens de textos escolhidos arbitrariamente

por um oráculo, e envia um par de textos para um desafiador que cifra um destes textos escolhidos aleatoriamente, e envia este texto cifrado para o atacante, que por sua vez pode continuar cifrando e decifrando textos (exceto o texto desafio) utilizando o oráculo para deduzir com uma probabilidade maior que 50% qual dos textos foi cifrado.

No caso do ML-KEM, a equipe CRYSTALS e o NIST, o classificaram como sendo IND-CCA2 seguro (AVANZI et al., 2012), isto significa que mesmo que fornecidos um oráculo de decifragem para um atacante, e permitir que ele realize cifragens e decifragens adaptativamente, ele não ira conseguir recuperar nenhuma informação de uma mensagem cifrada.

Outra classificação para um ataque de criptoanálise segundo (SCHNEIER, 1996) é com base na quantidade de recursos computacionais necessários, como a quantidade de processamento, armazenamento e dados para realizar o ataque. Embora exista um método capaz de resolver um problema computacional utilizado em um sistema criptográfico, ou recuperar informações úteis, isso não significa que este ataque seja viável computacionalmente. Por exemplo, é possível descobrir uma senha cifrada por uma função *hash* através de um ataque por força bruta, em que são testadas todas as combinações possíveis, entretanto analisar todas as possibilidades levaria uma quantidade de tempo muito grande. Dessa forma, mesmo que exista uma forma de recuperar a senha por força bruta, esta função *hash* é considerada segura contra este ataque devido ao tempo de processamento necessário. Existem diversos métodos que quebram o ML-KEM, mas todos eles demandam muitos recursos computacionais para recuperar alguma informação útil do texto original (AVANZI et al., 2012).

2.7 Considerações finais

Este capítulo tem a finalidade de fornecer informações essenciais sobre criptografia para que o leitor possa entender o funcionamento do algoritmo ML-KEM e assuntos relacionados. São apresentados os principais tipos de criptografia, os problemas da fatoração prima e do logaritmo discreto, estes empregados nos principais algoritmos de criptografia atuais e ameaçados pelo algoritmo de Shor. No Capítulo 3, é apresentada a estrutura de reticulados, na qual são estabelecidos problemas matemáticos de difícil resolução até mesmo por

computadores quânticos, utilizados em algoritmos de criptografia pós-quânticos baseados nesta estrutura.

3 Reticulados

Reticulados são objetos geométricos que podem ser descritos como pontos de uma grade regular n-dimensional (MICCIANCIO; GOLDWASSER, 2002). Esta estrutura tem sido alvo de pesquisadores da área da criptografia devido a problemas computacionais de difícil resolução tanto para computadores convencionais quanto para computadores quânticos. Tais problemas computacionais foram utilizados na construção de sistemas criptográficos como ML-KEM (STANDARDS; TECHNOLOGY, 2023), Dilithium (DUCAS et al., 2021), NTRU (CHEN et al., 2019) e FALCON (FOUQUE et al., 2020). Este capítulo é dedicado ao estudo e aplicabilidades dos reticulados na área da criptografia.

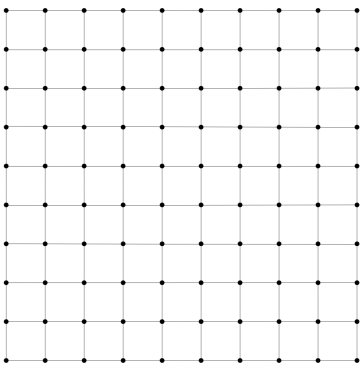
Definição 1 (reticulado)**.** *Um reticulado $\mathcal{L}$ é um subgrupo aditivo discreto de $\mathbb{R}^n$, ou seja, satisfaz as seguintes propriedades:*

(i) fechado para adição: $\forall x, y \in \mathcal{L},\ x + y \in \mathcal{L}$;

(ii) discreto: $\forall x, y \in \mathcal{L}$, em que $x \neq y$, existe um valor $\epsilon > 0$ tal que a distância entre eles é $\|x - y\| \geq \epsilon$.

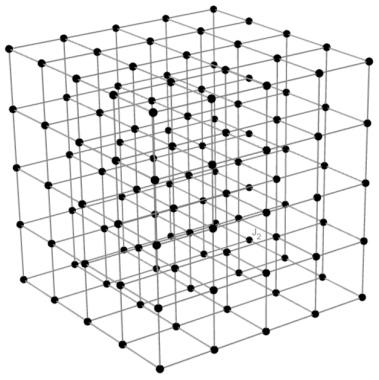
As Figuras 3.1 e 3.2 ilustram um exemplo de um reticulado de duas dimensões e três dimensões respectivamente.

Figura 3.1: Reticulado de duas dimensões.



Fonte: O autor.

Figura 3.2: Reticulado de três dimensões.



Fonte: O autor.

3.1 Base de um reticulado

Assim como os espaços vetoriais podem ser representados/gerados por uma combinação linear entre vetores linearmente independentes, os reticulados também, com exceção que os coeficientes da combinação linear devem pertencer ao conjunto $\mathbb{Z}$, o que faz com que esta estrutura seja regular (MICCIANCIO; GOLDWASSER, 2002). A definição de um reticulado $\mathcal{L}(\beta)$, gerado por uma base $\beta = \{b_0, b_1, ..., b_{n-1}\} \subset \mathbb{R}^n$, é expressa por:

$$\mathcal{L}(\beta) = \left\{ \sum_{i=0}^{n-1} m_i b_i \mid m_i \in \mathbb{Z},\ b_i \in \beta \right\},$$

Um elemento de um reticulado, por sua definição, é representado por um vetor ou ponto. Entretanto, devido aos isomorfismos aditivos σ_1 e σ_2, é possível representar os elementos de um reticulado na forma polinomial ou matricial respectivamente. A representação de um elemento na forma polinomial possui mais utilidade em alguns problemas específicos envolvendo reticulados como *Ring Learning With Errors* (RLWE) e MLWE.

$$\begin{aligned} \sigma_1 &: P[x] \to \mathbb{Z}^n \\ &: a_0 + a_1 x + a_2 x^2 + ... + a_{n-1} x^{n-1} \to (a_0, a_1, a_2, ..., a_{n-1}) \end{aligned}$$

$$\begin{aligned} \sigma_1^{-1} &: \mathbb{Z}^n \to P[x] \\ &: (a_0, a_1, a_2, ..., a_{n-1}) \to a_0 + a_1 x + a_2 x^2 + ... + a_{n-1} x^{n-1} \end{aligned}$$

Através do isomorfismo σ_1 é possível transformar um polinômio na forma $a_0 + a_1 x + a_2 x^2 + ... + a_{n-1} x^{n-1}$ em um ponto ou vetor na forma $(a_0, a_1, a_2, ..., a_{n-1})$. O inverso desse isomorfismo, representado por σ_1^{-1}, realiza o mapeamento inverso, transformando um ponto em um polinômio. Por outro lado, a representação de um elemento de um reticulado na forma matricial facilita a visualização na realização dos cálculos dos principais problemas computacionais sobre esta estrutura.

$$\begin{aligned} \sigma_2 &: M_n \to \mathbb{Z}^n \\ &: [a_0\ a_1\ a_2\ ...\ a_{n-1}] \to (a_0, a_1, a_2, ..., a_{n-1}) \end{aligned}$$

$$\begin{aligned} \sigma_2^{-1} &: \mathbb{Z}^n \to M_n \\ &: (a_0, a_1, a_2, ..., a_{n-1}) \to [a_0\ a_1\ a_2\ ...\ a_{n-1}] \end{aligned}$$

O isomorfismo aditivo σ_2 e sua inversa σ_2^{-1}, por sua vez, transformam uma matriz linha ou coluna em um elemento de $\mathbb{Z}^n$ e vice-versa. Estes isomorfismos aditivos significam que um elemento de um reticulado representado por um ponto, vetor, matriz ou polinômio, possui o mesmo comportamento com relação à operação de adição.

Uma base $\beta = \{b_0, b_1, ...b_{n-1}\} \subset \mathbb{R}^n$ de um reticulado também pode ser representada na forma matricial devido ao isomorfismo σ_2, na qual as componentes dos vetores da base formam as colunas da matriz geradora do reticulado. Seja $\mathbf{B} = [b_0\ b_1\ ...\ b_{n-1}] \in \mathbb{R}^{d\times n}$ a base do reticulado $\mathcal{L}$ na forma matricial, $\mathcal{L}(\mathbf{B})$ pode ser representada por:

$$\mathcal{L}(\mathbf{B}) = \{\mathbf{B}x \mid x \in \mathbb{Z}^n\},$$

em que $\mathbf{B}x$ é uma multiplicação matricial usual, d representa o número de componentes dos vetores de $\mathbf{B}$ e n representa a dimensão do reticulado $\mathcal{L}(\mathbf{B})$. Quando o número de componentes dos vetores da base de um reticulado coincide com seu grau, ou seja, a base do reticulado na forma matricial possui posto completo[1], o reticulado é dito completo.

Ambas as formas de representação de um reticulado são equivalentes, isso pode ser verificado derivando a notação por combinação linear a partir da notação matricial:

$$\begin{bmatrix} b_{00} & b_{01} & . & . & . & b_{0(n-1)} \\ b_{10} & b_{11} & . & . & . & b_{1(n-1)} \\ b_{20} & b_{21} & . & . & . & b_{2(n-1)} \\ . & . & . & & & . \\ . & . & & . & & . \\ . & . & & & . & . \\ b_{(d-1)0} & b_{(d-1)1} & & & & b_{(d-1)(n-1)} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ . \\ . \\ . \\ x_{n-1} \end{bmatrix} \Leftrightarrow$$

[1]Em outras palavras: tem-se uma matriz quadrada.

$$x_0 \begin{bmatrix} b_{00} \\ b_{01} \\ b_{02} \\ . \\ . \\ . \\ b_{0(d-1)} \end{bmatrix} + x_1 \begin{bmatrix} b_{10} \\ b_{11} \\ b_{12} \\ . \\ . \\ . \\ b_{1(d-1)} \end{bmatrix} + x_2 \begin{bmatrix} b_{20} \\ b_{21} \\ b_{22} \\ . \\ . \\ . \\ b_{2(d-1)} \end{bmatrix} + ... + x_{n-1} \begin{bmatrix} b_{(n-1)0} \\ b_{(n-1)1} \\ b_{(n-1)2} \\ . \\ . \\ . \\ b_{(n-1)(d-1)} \end{bmatrix} \Leftrightarrow$$

$$x_0\vec{b_0} + x_1\vec{b_1} + x_2\vec{b_2} + ... + x_{n-1}\vec{b_{n-1}}$$

Existem alguns tipos de reticulados que devem ser mencionados para melhor compreensão dos conceitos utilizados posteriormente neste trabalho.

Definição 2 (reticulado inteiro)**.** *Um reticulado $\mathcal{L}(\boldsymbol{B})$ é dito inteiro se os coeficientes dos vetores da base $\boldsymbol{B}$ forem inteiros, ou seja, $\mathcal{L}(\boldsymbol{B}) = \{\boldsymbol{B}x \mid \boldsymbol{B} \in \mathbb{Z}^{d\times n},\ x \in \mathbb{Z}^n\}$.*

Definição 3 (reticulado dual)**.** *A dual de um reticulado $\mathcal{L}$, denotado por $\mathcal{L}^*$, é o conjunto de todos os vetores $\vec{x} \in span(\mathcal{L})$ tal que $\langle \vec{x}, \vec{y} \rangle \in \mathbb{Z}$, para todo $\vec{y} \in \mathcal{L}$.*

Definição 4 (reticulado ideal)**.** *Seja um anel A e um isomorfismo aditivo σ que mapeia A para $\sigma(A) \subset \mathbb{R}^n$. Um reticulado ideal é um $\sigma(I)$ para um ideal $I \subseteq A$ (LYUBASHEVSKY; PEIKERT; REGEV, 2010).*

Segundo a Definição 4, um reticulado ideal é obtido aplicando um isomorfismo aditivo que mapeia um ideal para um subconjunto de $\mathbb{R}^n$. A estrutura $\mathbb{Z}[x]/\langle x^n + 1\rangle$, na qual é abordada na Seção 3.4 e pode ser consultada no Apêndice A, representa a um reticulado ideal, visto que existe um isomorfismo aditivo que mapeia um polinômio pertencente ao ideal $\langle x^n + 1\rangle$ para um subconjunto de $\mathbb{R}^n$, neste caso $\mathbb{Z}^n$.

Definição 5 (reticulado modular)**.** *Seja um anel A e um isomorfismo aditivo σ que mapeia A para $\sigma(A) \subset \mathbb{R}^n$. Um reticulado modular é definido por uma função que mapeia um conjunto $\{\sigma(A)_1, \sigma(A)_2, ..., \sigma(A)_d\}$ para $\mathbb{R}^{nd}$ (LANGLOIS; STEHLE, 2012).*

Um reticulado modular é um subconjunto de um espaço vetorial gerado por um conjunto finito de vetores fechado sob operações de soma e multiplicação por escalares em um anel comutativo. Tem-se como exemplo uma matriz de ordem $n \times n$ em que os elementos desta matriz pertencem a um anel. Se houver um isomorfismo que mapeia os elementos desta matriz para um subconjunto de $\mathbb{R}^n$ e que esta matriz respeita as propriedades de módulo, então pode-se afirmar que esta matriz pertence a um módulo.

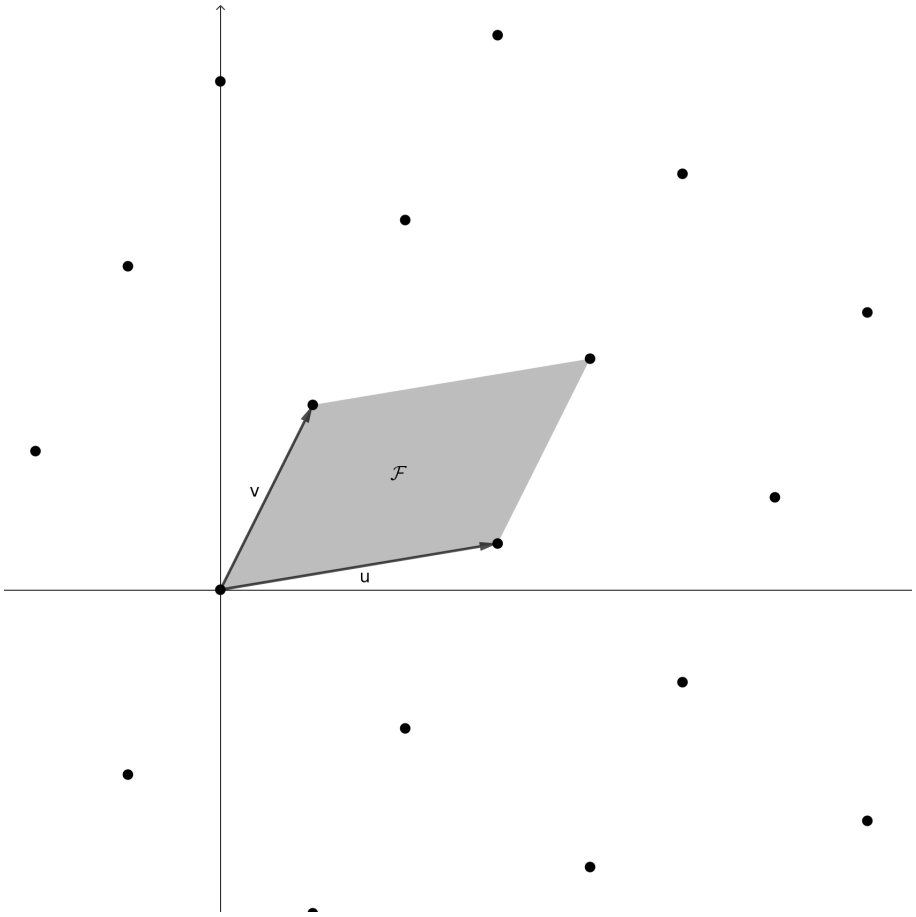
3.2 Domínio fundamental

O domínio fundamental de um reticulado $\mathcal{L}(\mathbf{B})$, denotado por $\mathcal{F}_B$, é a região do paralelepípedo de dimensão n formado pelos vetores geradores de $\mathcal{L} \subset \mathbb{R}^n$. Tal paralelepípedo é expresso geometricamente pela fórmula abaixo:

$$\mathcal{F}_B = \{\mathbf{B}x \mid 0 \leq x_i < 1\},$$

na qual x_i são as componentes do vetor x, como pode ser visto na Figura 3.3.

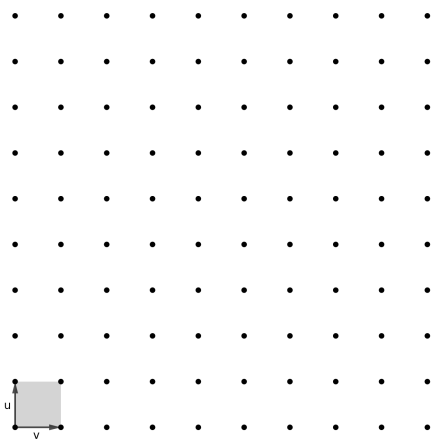
Figura 3.3: Ilustração de um domínio fundamental.



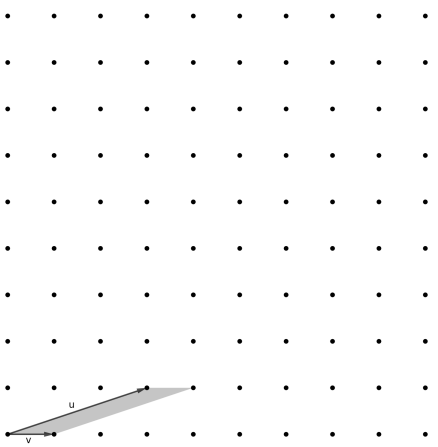
Fonte: O autor.

As regiões sombreadas da Figura 3.4 representam os domínios fundamentais de um mesmo reticulado de dimensão 2 gerado por duas bases diferentes, na qual as áreas dos dois paralelepípedos são as mesmas e nenhum elemento do reticulado está no domínio fundamental, com exceção da origem. Isto significa que ambas as bases geram este mesmo reticulado. Esta é uma das propriedades dos domínios fundamentais segundo (MICCIANCIO; GOLDWASSER, 2002).

Figura 3.4: Exemplo de domínios fundamentais diferentes com mesma área.



Fonte: O autor.



Fonte: O autor.

Sendo $\mathcal{L}$ um reticulado gerado por uma base $\mathbf{B}$, o volume de seu domínio fundamental pode ser calculado pelo valor absoluto da determinante de $\mathbf{B}$ na forma matricial, isto é:

$$Vol(\mathcal{F}_B) = |Det(\mathbf{B})|$$

e também através do algoritmo de Gram Schmidt visto na Seção 3.6 por meio da seguinte equação segundo (BARROS, 2014):

$$Vol(\mathcal{F}_B) = \prod_{i=1}^{n} \|b_i^*\|$$

em que B é a base de um reticulado de dimensão n e b_i^* são os vetores gerados pelo algoritmo de Gram Schmidt da base B. Embora a base retornada pelo algoritmo de Gram Schmidt não gere o mesmo reticulado da base da entrada, ambas possuem o mesmo volume.

3.3 Sucessivas mínimas

Seja $B_n(0, r) = \{x \in \mathbb{R}^n \mid ||x|| < r\}$ uma bola de dimensão n centrada na origem e raio r. As sucessivas mínimas $\lambda_1, \lambda_2, ..., \lambda_n$ de um reticulado $\mathcal{L}$ de dimensão n são constantes, onde $\lambda_i(\mathcal{L})$ é o raio da menor bola de dimensão i centrada na origem que contém i vetores linearmente independentes de $\mathcal{L}$. As sucessivas mínimas de um reticulado $\mathcal{L}(\mathbf{B})$

são definidas da seguinte forma:

$$\lambda_i(\mathcal{L}) = min\{r \mid dim(\mathcal{L}(\mathbf{B}) \cap \mathcal{B}(0,r)) \geq i\}$$

Tem-se na Figura 3.5 um reticulado de duas dimensões, isso significa que este reticulado tem apenas duas sucessivas mínimas λ_1 e λ_2. Analisando este reticulado da Figura 3.5, a bola de dimensão 1 de menor raio e que contém um vetor linearmente dependente deste reticulado está destacada em uma cor mais escura e possui raio λ_1, e a bola de dimensão 2 de menor raio que contém os vetores linearmente independentes deste reticulado está destacada em uma cor mais clara e possui raio λ_2.

Uma propriedade das sucessivas mínimas é que elas satisfazem $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$. Desta forma, λ_1 representa o tamanho do menor vetor de um reticulado com exceção do vetor nulo e também a menor distância entre dois elementos distintos desse reticulado (MICCIANCIO; GOLDWASSER, 2002).

$$\lambda_1(\mathcal{L}) = \min_{x \neq y \in \mathcal{L}} ||x - y|| = \min_{x \in \mathcal{L} - \vec{0}} ||x||$$

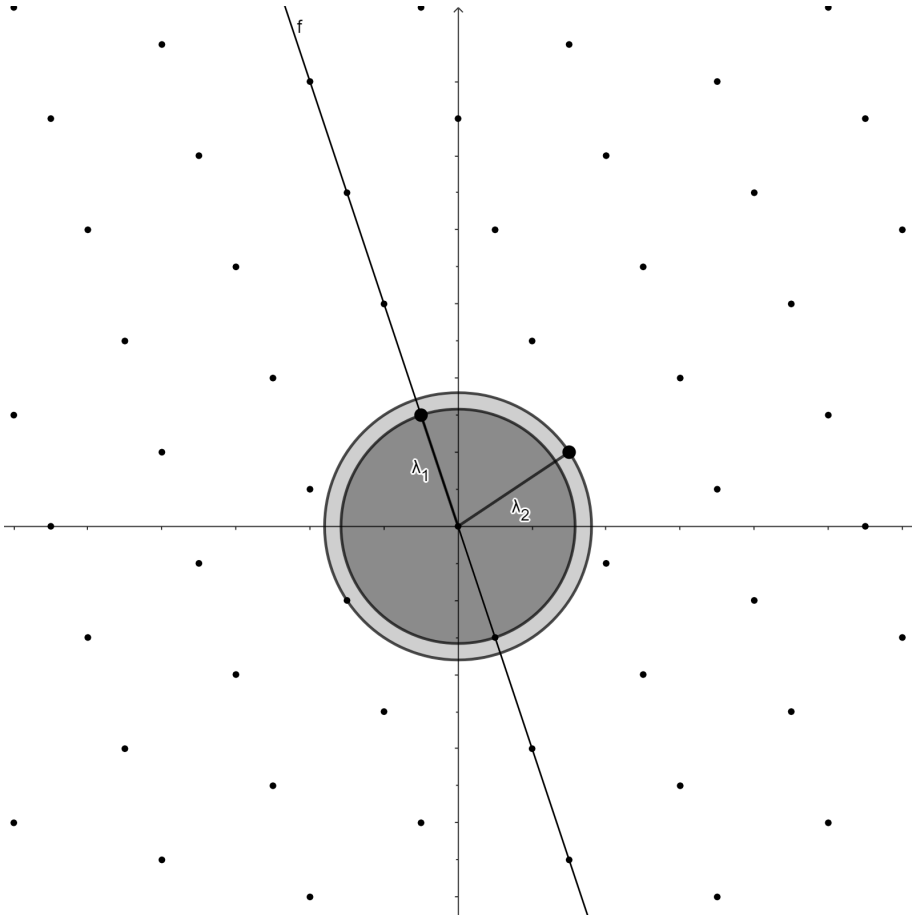
Alguns problemas baseados em reticulados estão relacionados com o menor vetor de um reticulado, também denotado por λ_1.

As sucessivas mínimas de um reticulado dependem da métrica utilizada. Sejam os vetores $\vec{b_1} = \{2, 0\}$ e $\vec{b_2} = \{1, 1\}$, usando a métrica l_1 definida na Seção A.5, $\lambda_1 = \vec{b_1}$, visto que a norma de $\vec{b_1}$ sob a métrica l_1 é 2. Entretanto, $\lambda_1 \neq \vec{b_1}$ se utilizadas as normas l_2 ou l_∞ por exemplo. Isso se dá devido a $\vec{b_2}$ ser menor que $\vec{b_1}$ quando usadas essas métricas (MICCIANCIO; GOLDWASSER, 2002). Dessa forma, ao realizar uma análise de um problema computacional envolvendo sucessivas mínimas, deve-se ter bem definida qual métrica está sendo utilizada.

3.4 Problemas computacionais

Nesta seção serão apresentados alguns dos principais problemas computacionais sobre a estrutura de reticulados que formam a base da segurança dos sistemas de criptografia que utilizam esta estrutura. Como será visto nas Seções 3.5 e 3.6, os algoritmos que resolvem estes problemas possuem complexidade exponencial ou pior, e os que executam em tempo

Figura 3.5: Sucessivas mínimas de um reticulado de duas dimensões.



Fonte: O autor.

polinomial conseguem apenas aproximações distantes do resultado ótimo para reticulados com certas propriedades, como dimensão, ortogonalidade e tamanho da norma dos vetores que formam a base do reticulado. Devido a este fato, os algoritmos que se baseiam nesses problemas são considerados seguros.

Definição 6 (*Closest Vector Problem* (CVP))**.** *O problema CVP consiste em, dado um reticulado* $\mathcal{L}$ *e um vetor* $\vec{c} \notin \mathcal{L}$*, encontrar um vetor* $\vec{v} \in \mathcal{L}$ *tal que a distância entre* $\vec{c}$ *e* $\vec{v}$ *seja mínima.*

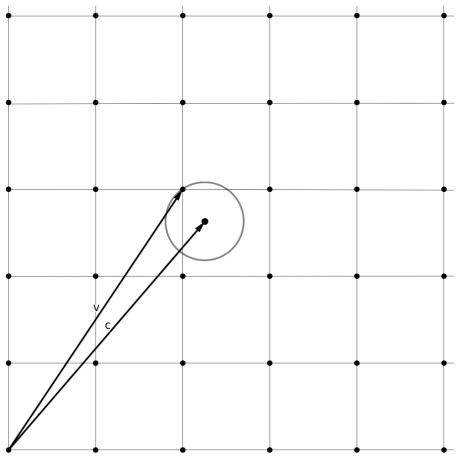
Intuitivamente, o problema CVP pode ser aplicado à criptografia da seguinte maneira, seja $\vec{v} \in \mathcal{L}(\beta)$ uma chave privada, é somado a esta chave um vetor aleatório $\vec{e} \notin \mathcal{L}(\beta)$ com norma pequena, o resultado dessa soma é uma chave pública $\vec{c} \notin \mathcal{L}(\beta)$, dessa forma, para se recuperar a chave privada é preciso encontrar o vetor mais próximo de $\vec{c}$, que nesse caso seria a chave privada $\vec{v}$. Este é um dos principais problemas envolvendo reticulados e foi demonstrado por (BOAS, 1981) pertencer à classe dos problemas *Non-deterministic Polynomial-time Hardness* (NP-HARD) para todas as normas l_p. A Figura

3.6 ilustra o problema CVP em um reticulado de duas dimensões.

Definição 7 (*Shortest Vector Problem* (SVP))**.** *O problema SVP consiste em, dado um reticulado $\mathcal{L}$, encontrar um vetor $\vec{v} \in \mathcal{L}$ tal que $\|\vec{v}\|$ seja mínima.*

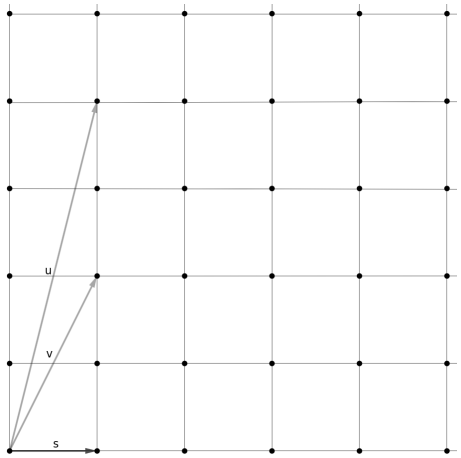
Outro problema importante associado a reticulados é o problema SVP, diferente do problema CVP, este consiste em encontrar o vetor mais próximo da origem, que seria no caso o menor vetor do reticulado. Devido a essa relação com o problema CVP, existe uma redução do problema SVP para o CVP, demonstrada em (GOLDREICH et al., 1999). O problema SVP foi demonstrado pertencer à classe NP-HARD para a métrica l_∞ por (BOAS, 1981). A Figura 3.7 ilustra este problema, em que w é o vetor mais curto do reticulado.

Figura 3.6: Ilustração do problema CVP.



Fonte: O autor.

Figura 3.7: Ilustração do problema SVP.



Fonte: O autor.

Definição 8 (CVP_γ)**.** *O problema CVP_γ consiste em dado um reticulado $\mathcal{L}$, um vetor $\vec{c} \notin \mathcal{L}$ e um fator de aproximação $\gamma \geq 1$, encontrar um vetor $v \in \mathcal{L}$ tal que $\|\vec{c} - \vec{v}\| \leq \gamma d(\vec{c}, \mathcal{L})$.*

Definição 9 (SVP_γ)**.** *O problema SVP_γ consiste em, dado um reticulado $\mathcal{L}$ e um fator de aproximação $\gamma \geq 1$, encontrar um vetor $\vec{v} \in \mathcal{L}$ tal que $0 < \|\vec{v}\| \leq \gamma\lambda_1(\mathcal{L})$.*

Os problemas que utilizam um fator de aproximação, como CVP_γ e SVP_γ, são estudados nos algoritmos que conseguem apenas uma aproximação do resultado ótimo para estes problemas. Esse fator geralmente está em função do grau do reticulado, por exemplo $\gamma(n) = (2/\sqrt{3})^n$, isso significa que o resultado desse algoritmo estará no máximo a γ de distância do resultado ótimo. Os algoritmos que resolvem uma aproximação desses problemas se encontram na Seção 3.6.

Definição 10 (*Gap Shortest Vector Problem* ($GapSVP_\gamma$)). *O problema $GapSVP_\gamma$ consiste em, dado um reticulado $\mathcal{L}$ e um número real positivo d, determinar se $\lambda_1(\mathcal{L}) \leq d$ ou $\lambda_1(\mathcal{L}) > \gamma d$.*

Definição 11 (*Shortest Independent Vector Problem* ($SIVP_\gamma$)). *O problema $SIVP_\gamma$ consiste em, fornecido um reticulado $\mathcal{L}$ de dimensão n, encontrar um conjunto de n vetores linearmente independentes com comprimento máximo de $\gamma\lambda_n(\mathcal{L})$.*

Definição 12 (*Learning With Errors* (LWE)). *Dado uma matriz $\boldsymbol{A} \in \mathbb{Z}_p^{m\times n}$ gerada por uma distribuição de probabilidade uniforme e uma matriz $\boldsymbol{t} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e} \in \mathbb{Z}_p^m$, sendo $\boldsymbol{e} \in \mathbb{Z}_p^m$ um erro adicional especificado por uma distribuição de probabilidade $\chi : \mathbb{Z}_p \to \mathbb{R}^+$ em $\mathbb{Z}_p$, encontrar a matriz $\boldsymbol{s} \in \mathbb{Z}_p^n$ (REGEV, 2009).*

Figura 3.8: Ilustração do problema LWE.

$$\begin{bmatrix} a_{00} & a_{01} & \cdots & a_{0(n-1)} \\ a_{10} & a_{11} & \cdots & a_{1(n-1)} \\ a_{20} & a_{21} & \cdots & a_{2(n-1)} \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ a_{(m-1)0} & a_{(m-1)1} & & a_{(m-1)(n-1)} \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ \vdots \\ \vdots \\ \vdots \\ s_n \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ \vdots \\ \vdots \\ \vdots \\ e_n \end{bmatrix} = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \\ \vdots \\ \vdots \\ \vdots \\ t_n \end{bmatrix}$$

Fonte: O autor.

O problema LWE foi introduzido por (REGEV, 2009) e consiste na dificuldade de se obter a solução para uma equação linear com um erro, como pode ser observado na Figura 3.8. Perceba que sem a matriz de erro/ruído **e**, este problema poderia ser facilmente resolvida pela eliminação Gaussiana em tempo $\mathcal{O}(n)$, sendo n o número de equações a serem resolvidas. Entretanto, se adicionarmos um erro aleatório a essa equação, resolvê-la é uma tarefa difícil, até para computadores quânticos. O artigo (REGEV, 2009) relata a dificuldade de se encontrar algoritmos clássicos e quânticos que resolvem este problema em tempo polinomial, o que o torna útil para a criação de sistemas de criptografia baseados neste problema. A complexidade de resolução do LWE está relacionada com a dificuldade de resolução dos problemas *Gap Shortest Vector Problem* (GapSVP) e *Shortest Independent Vector Problem* (SIVP), visto que (REGEV, 2009) demonstra uma redução destes problemas para LWE.

Este problema, além de ser seguro contra computadores quânticos, chamou atenção devido a sua versatilidade, eficiência e simplicidade. Ele foi a base para a criação de outros problemas computacionais como RLWE, MLWE e também outros modelos de criptografia, como criptografia homomórfica.

Um aspecto a ser considerado ao utilizar o LWE é o tamanho da chave pública, representado pela matriz $\mathbf{A}$. Em que a complexidade de espaço é $\mathcal{O}(n^2)$, sendo n a ordem da matriz $\mathbf{A}$. Esta é uma das razões pela qual este problema não é utilizado na prática em sistemas criptográficos. O problema *Ring Learning With Errors* (RLWE) apresenta uma alternativa para diminuir o tamanho da matriz $\mathbf{A}$ e melhorar o desempenho das operações de multiplicação matricial devido à seguinte relação:

$$\begin{bmatrix} b_1 & -b_n & -b_{n-1} & . & . & . & -b_2 \\ b_2 & b_1 & -b_{n-2} & . & . & . & -b_3 \\ b_3 & b_2 & b_1 & . & . & . & -b_4 \\ . & . & & . & . & . & . \\ . & . & & & . & . & . \\ . & . & & & & . & . \\ b_n & b_{n-1} & & & & & b_1 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ . \\ . \\ . \\ s_n \end{bmatrix} \Leftrightarrow$$

$$(b_1 + b_2x + b_3x^2 + ... + b_nx^{n-1})(s_1 + s_2x + s_3x^2 + ... + s_nx^{n-1}) \textbf{ mod } x^n + 1$$

Dessa forma é possível obter os demais elementos da matriz a partir da primeira coluna, sendo assim, necessário apenas armazenar a primeira coluna, o que resulta em uma diminuição do tamanho da chave criptográfica.

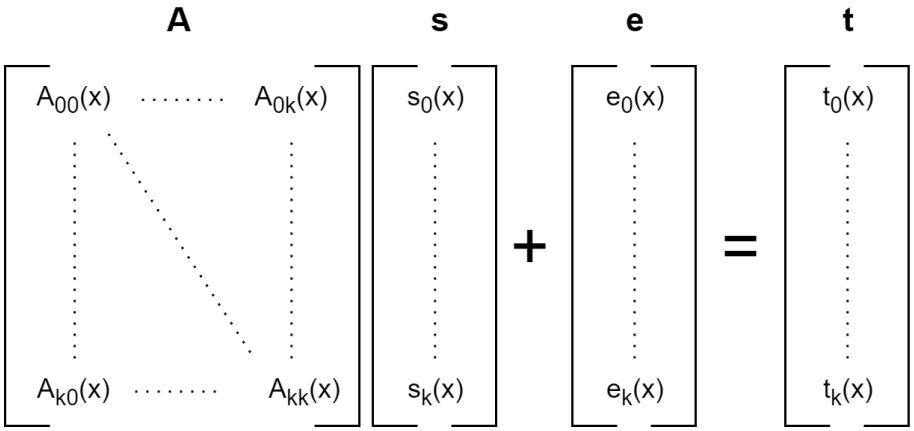
Definição 13 (*Ring Learning With Errors* (RLWE))**.** *Seja $A \in \mathbb{Z}_q[x]/\langle x^n+1\rangle$ gerada por uma distribuição de probabilidade uniforme e um polinômio $t = As + e \in \mathbb{Z}_q[x]/\langle x^n + 1\rangle$ sendo $e \in \mathbb{Z}[x]/\langle x^n+1\rangle$ especificado por uma distribuição de probabilidade χ em $\mathbb{Z}[x]/\langle x^n+1\rangle$ com desvio padrão σ, considerando-o em $\mathbb{Z}_q[x]/\langle x^n + 1\rangle$, encontrar o polinômio $s \in \mathbb{Z}_q[x]/\langle x^n + 1\rangle$ (LYUBASHEVSKY; PEIKERT; REGEV, 2010).*

A redução do tamanho da chave não é a única vantagem oferecida pelo RLWE, transformando os vetores $\vec{b}$ e $\vec{s}$ em polinômios através do isomorfismo $\sigma_1 : P[x] \to \mathbb{Z}^n$, a multiplicação entre $\sigma_1^{-1}(\vec{b})$ e $\sigma_1^{-1}(\vec{s})$ módulo $x^n + 1$ é equivalente à multiplicação matricial acima. A realização de uma operação de multiplicação polinomial modular, em vez da matricial, proporciona aceleração à operação de multiplicação por meio do algoritmo

Number Theoretic Transform (NTT) (SALARIFARD; SOLEIMANY, 2023), onde a complexidade passa a ser $\mathcal{O}(nlog(n))$. A redução do tamanho da chave, segundo (LYUBASHEVSKY; PEIKERT; REGEV, 2010), não reduz a complexidade do problema, desde que se escolha os parâmetros σ, n e q adequadamente.

Definição 14 (*Module Learning With Errors* (MLWE))**.** *Seja* $\boldsymbol{A} \in [\mathbb{Z}_q[x]/\langle x^n+1\rangle]^{m\times k}$ *gerada por uma distribuição de probabilidade uniforme e uma matriz* $\boldsymbol{t} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e} \in [\mathbb{Z}_q[x]/\langle x^n+1\rangle]^m$ *sendo* $\boldsymbol{e} \in [\mathbb{Z}_q[x]/\langle x^n+1\rangle]^m$*, encontrar a matriz* $\boldsymbol{s} \in [\mathbb{Z}_q[x]/\langle x^n+1\rangle]^k$ *(LANGLOIS; STEHLE, 2012).*

Figura 3.9: Ilustração do problema MLWE.



Fonte: O autor.

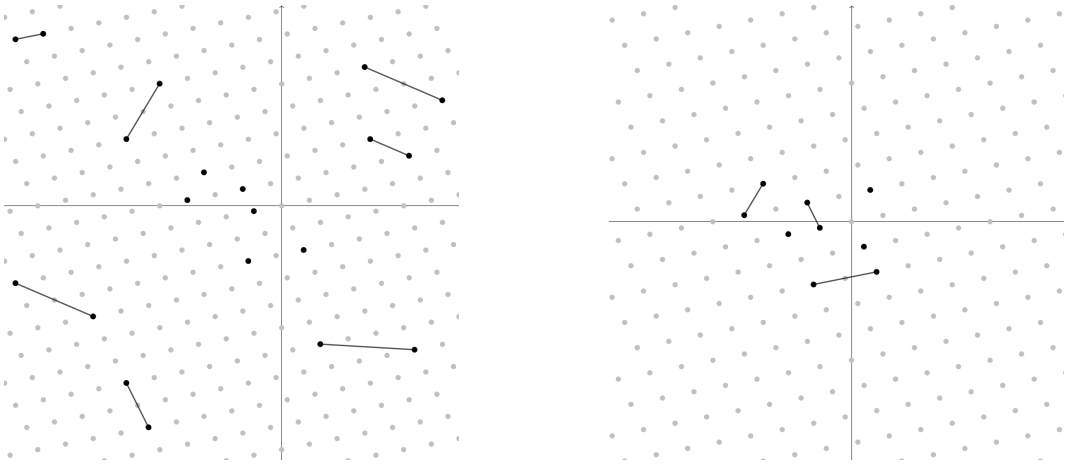
O problema MLWE é semelhante ao problema LWE como pode ser observado na Figura 3.9, com a diferença que os elementos das matrizes, em vez de inteiros, agora são polinômios. O MLWE pode ser interpretado como vários problemas RLWE dentro do LWE, visto que se alterado o parâmetro k para 1, tem-se exatamente o problema RLWE. Segundo a documentação oficial do ML-KEM (STANDARDS; TECHNOLOGY, 2023), os autores optaram por utilizar o MLWE por preocupações que a estrutura RLWE possa permitir ataques mais eficientes, além de que o MLWE possui uma melhor escalabilidade e eficiência semelhante ao RLWE para determinados parâmetros (AVANZI et al., 2012).

3.5 Algoritmos exatos

Existem duas técnicas que resolvem com exatidão os problemas SVP e CVP, a técnica de *sieving* e a técnica de *enumeration*.

A técnica de *sieving* foi proposta pela primeira vez em 2001 por Miklós Ajtai, Ravi Kumar e D. Sivakumar (AJTAI; KUMAR; SIVAKUMAR, 2001), e existem diversas variações deste algoritmo com complexidade de tempo inferior ao algoritmo original $\mathcal{O}(2^n)$, porém ainda com complexidade exponencial de tempo e espaço, o que torna estes algoritmos impraticáveis para reticulados de grandes dimensões. A ideia principal desta técnica é selecionar um conjunto de vetores em uma região de raio R de um reticulado, realizar sucessivas subtrações entre pares de vetores deste conjunto e adicionar os vetores resultantes dessas operações ao conjunto final caso sua norma for menor que γR para $0 < \gamma < 1$. A Figura 3.10 ilustra a ideia-chave dos algoritmos que utilizam a técnica de *sieving*.

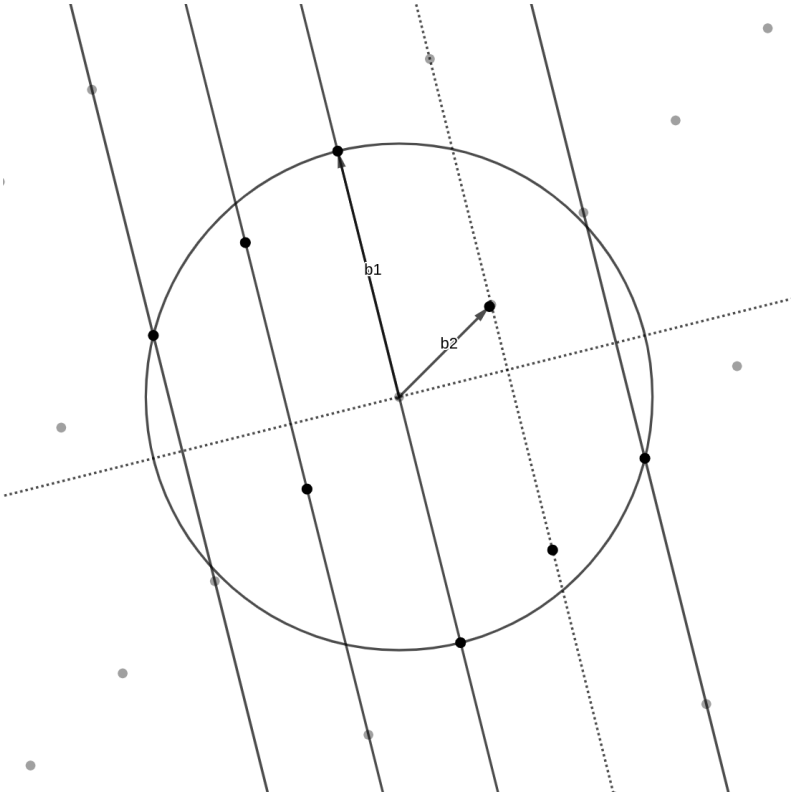
Figura 3.10: Exemplo do funcionamento do algoritmo *sieving* com duas iterações respectivamente.



Fonte: O autor.

A técnica de *enumeration* foi proposta pela primeira vez por U. Fincke e M. Pohst em 1985 (FINCKE; POHST, 1985), entretanto, devido à sua complexidade de $\mathcal{O}(2^{n^2})$, é mais conhecido o algoritmo proposto por Ravi Kannan de 1987, com complexidade de $\mathcal{O}(n^n)$ (KANNAN, 1987). Este algoritmo, assim como o AKS, também resolve o problema SVP com exatidão, e embora possua complexidade de tempo maior que o algoritmo AKS $\mathcal{O}(n^n)$, este algoritmo tem a vantagem de possuir complexidade polinomial de espaço. A ideia principal dos algoritmos de *enumeration* é selecionar duas bases $\vec{b_1}$ e $\vec{b_2}$, calcular a projeção de $\vec{b_2}$ sobre a reta ortogonal a $\vec{b_1}$. O próximo passo é enumerar os vetores em uma região que estão a uma distância múltipla da norma do vetor resultante da projeção de $\vec{b_2}$ sobre a reta ortogonal a $\vec{b_1}$. A partir destes vetores enumerados o algoritmo de Kannan encontra o vetor mais curto nesta região. A Figura 3.11 ilustra este processo, onde os pontos destacados são os vetores enumerados pelo algoritmo.

Figura 3.11: Exemplo do funcionamento do algoritmo *enumeration*.



Fonte: O autor.

3.6 Redução de base

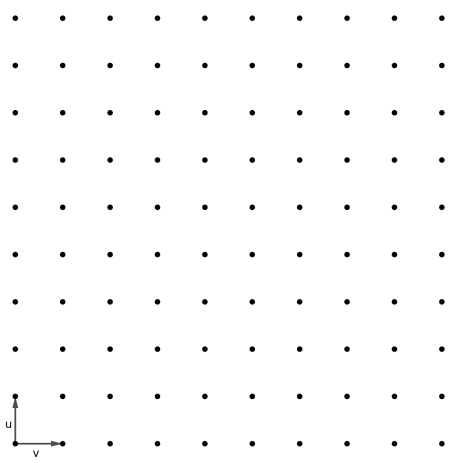
A segurança dos algoritmos baseados em problemas envolvendo reticulados está relacionada com algumas propriedades da base do reticulado, mais especificamente a ortogonalidade, norma e dimensão dos vetores da base. A técnica de redução de base de um reticulado consiste em encontrar outra base que gere este mesmo reticulado, mas com vetores com a menor norma possível e mais ortogonais entre si. É possível calcular a ortogonalidade da base de um reticulado pela *Razão de Hadamard*, esta é dada pela fórmula

$$\mathcal{H}(\beta) = \left(\frac{|Det(\mathbf{B})|}{\prod_{i=1}^{n} ||b_i||} \right)^{\frac{1}{n}}$$

em que $0 < \mathcal{H}(\mathbf{B}) \leq 1$(BARROS, 2014). Quanto mais próximo de 1 for $\mathcal{H}(\mathbf{B})$, mais ortogonal é a base $\mathbf{B}$, esta é dita uma base boa, enquanto para uma base ruim os valores de $\mathcal{H}(\mathbf{B})$ se aproximam de 0.

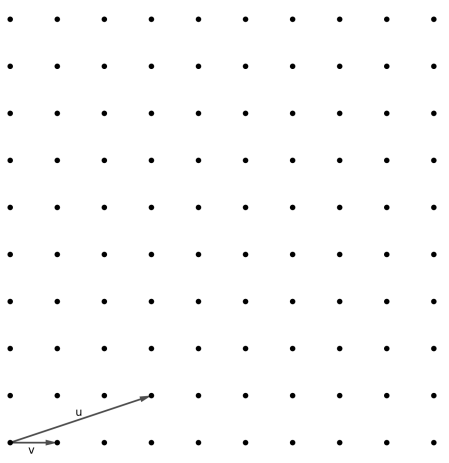
Os reticulados das Figuras 3.12 e 3.13, embora gerados por vetores completamente diferentes, são iguais tendo em vista que os vetores que geram o reticulado da Figura 3.13 podem ser expressos por uma combinação linear dos vetores que geram o reticulado

Figura 3.12: Exemplo de uma base boa.



Fonte: O autor.

Figura 3.13: Exemplo de uma base ruim.



Fonte: O autor.

da Figura 3.12. Entretanto, a base da Figura 3.12, gerada pelos vetores $\{(1,0),(0,1)\}$ é uma base boa, pois os vetores são ortogonais entre si e possuem sua norma pequena, a prova disso é o cálculo da *Razão de Hadamard* da base, que é exatamente 1. Já a base da Figura 3.13 é considerada ruim, visto que a *Razão de Hadamard* é aproximadamente 0,56. A base ser boa implica que conseguiremos boas aproximações pelos algoritmos de redução de base que serão abordados na seção 3.6.1.

3.6.1 Algoritmos de redução de base

A razão pela qual deseja-se uma base reduzida, é para se obter uma boa aproximação para o problema CVP através do algoritmo de Babai. O algoritmo de Babai (BABAI, 1985) proposto em 1985 por László Babai, realiza uma aproximação da solução do problema CVP dada uma base boa, na qual este funciona da seguinte maneira:

1. Sendo $\vec{s} \notin \mathcal{L}(\beta)$, escreva $\vec{s}$ como uma combinação linear de β;
2. Os coeficientes da combinação linear devem ser arredondados para um inteiro mais próximo;
3. Os coeficientes arredondados combinados com a base devem gerar uma aproximação do vetor mais próximo de $\vec{s}$ que pertence à $\mathcal{L}(\beta)$.

Esse algoritmo realiza uma boa aproximação do problema CVP ao ter uma base bem ortogonal, do contrário, o resultado não é preciso. Tenha como exemplo um reticulado

$\mathcal{L}(\beta)$ gerado pela base $\beta = \{(5,1),(-2,8)\}$ e um vetor $\vec{s} = (27,8) \notin \mathcal{L}(\beta)$. A *Razão de Hadamard* de β é $\mathcal{H}(\beta) \approx 0.999$, isso significa que devemos obter uma boa aproximação. Ao realizarmos os cálculos, obtém-se o vetor $\vec{c} = (30,6)$, sendo relativamente próximo de $\vec{s}$. Agora, aplicando o algoritmo de Babai para outra base $\beta = \{(37,41),(103,113)\}$ menos ortogonal que gera o mesmo reticulado, em que $\mathcal{H}(\beta) \approx 0.07$, e que desejamos encontrar o vetor mais próximo de $\vec{s} = (27,8) \notin \mathcal{L}(\beta)$ obtém-se a seguinte aproximação $\vec{c} = -53(37,41) + 19(103,113) = (-4,-26)$. Perceba que a aproximação é pouco precisa com relação à anterior.

Segundo (BARROS, 2014) o algoritmo de Babai enxerga um vetor que não pertence a um reticulado dentro de um domínio fundamental, onde o resultado deste algoritmo estará no domínio fundamental deste vetor. Desta forma, quanto menos ortogonal for um domínio fundamental, mais distante do resultado ótimo este algoritmo pode retornar.

Outro algoritmo importante é o de Gram-Schmidt, este algoritmo realiza sucessivas projeções entre pares de vetores da base com fim de retornar uma base com $\mathcal{H}(\beta) = 1$. Este algoritmo não retorna necessariamente uma base de vetores com coeficientes inteiros, visto que nem todo reticulado pode ser gerado por uma base totalmente ortogonal. Entretanto, este algoritmo é utilizado como subprocesso em outros algoritmos relacionados aos reticulados.

Algoritmo 3: Algoritmo de Gram-Schmidt

Entrada: $b_0, b_1, ..., b_n \in \mathbb{R}^m$

Saída: $b_0^*, b_1^*, ..., b_n^* \in \mathbb{R}^m$

```
b_0^* = b_0 / |b_0|
para i ← 1 até n faça
    para j ← 0 até i faça
        b_i ← b_i − ⟨b_i, b_j⟩ b_j
    b_i^* ← b_i / |b_i|
retorna b_0^*, b_1^*, ..., b_n^*
```

O algoritmo *Lenstra–Lenstra–Lovász* (LLL) (LENSTRA; LENSTRA; LOVÁSZ, 1982) é um dos principais algoritmos de redução de base de reticulados. O Algoritmo LLL realiza uma aproximação da solução ótima dos problemas SVP e CVP. Para o problema SVP o LLL realiza uma aproximação de $(2/\sqrt{3})^n \lambda_1$ em que n é a dimensão do

reticulado. Isto é, dada uma base, o algoritmo LLL irá encontrar um vetor que está a $(2/\sqrt{3})^n$ de distância máxima do vetor mais curto deste reticulado. E, para o CVP, o LLL irá encontrar um vetor que está a uma distância de $2(2/\sqrt{3})^n$ do vetor alvo (MICCIANCIO; GOLDWASSER, 2002). O Algoritmo 4 descreve o funcionamento do algoritmo LLL.

Algoritmo 4: Algoritmo LLL

Entrada: $b_0, b_1, ..., b_n \in \mathbb{Z}^m$, $\delta = \frac{3}{4}$

Saída: $b_0, b_1, ..., b_n \in \mathbb{Z}^m$

$\mu_{i,j} = \frac{b_i.b_j^*}{b_j^*.b_j^*}$

início

  $b_0^*, b_1^*, ..., b_n^* \leftarrow$ Gram-Schmidt$(b_0, b_1, ..., b_n)$

  $k \leftarrow 1$

  enquanto $k \leq n$ **faça**

    para $j \leftarrow k-1$ *até* 0 **faça**

      se $|\mu_{k,j}| > \frac{1}{2}$ **então**

        $b_k \leftarrow b_k - \lfloor \mu_{k,j} \rceil b_j$

        $b_0^*, b_1^*, ..., b_n^* \leftarrow$ Gram-Schmidt$(b_0, b_1, ..., b_n)$

    se $\langle b_k^*, b_k^* \rangle > (\delta - \mu_{k,k-1}^2)\langle b_{k-1}^*, b_{k-1}^* \rangle$ **então**

      $k \leftarrow k + 1$

    senão

      Swap$(b_k, b_k - 1)$

      $b_0^*, b_1^*, ..., b_n^* \leftarrow$ Gram-Schmidt$(b_0, b_1, ..., b_n)$

      $k \leftarrow Max(k-1, 1)$

  retorna $b_0, b_1, ..., b_n$

3.7 Considerações finais do capítulo

Este capítulo apresenta a estrutura de reticulados e seus conceitos fundamentais, como também os principais problemas matemáticos envolvendo esta estrutura e as técnicas e algoritmos que resolvem estes problemas ou encontram uma solução aproximada do resultado ótimo. Estes conceitos, problemas e algoritmos estão relacionados com o algoritmo ML-KEM apresentado no Capítulo 4.

4 ML-KEM

O ML-KEM (STANDARDS; TECHNOLOGY, 2023) é um sistema criptográfico assimétrico baseado na dificuldade de se resolver o problema MLWE sobre a estrutura algébrica de reticulados. ML-KEM é um dos algoritmos finalistas do programa PQC do NIST e selecionado para padronização. Neste capítulo iremos abordar o funcionamento do ML-KEM, sua segurança e ataques criptoanalíticos conhecidos contra esse criptossistema. Devido à falta de materiais de fácil compreensão sobre o ML-KEM e toda a fundamentação matemática relacionada, este capitulo irá abordar o algoritmo ML-KEM de uma maneira mais didática, reduzindo alguns parâmetros e removendo algumas funções que não alteram a estrutura do algoritmo, de forma a simplificar e facilitar o entendimento do leitor.

O ML-KEM é um algoritmo de encapsulamento de chave, e devido à estrutura deste mecanismo é possível separar este criptossistema em duas partes, a parte de cifragem assimétrica, denominada pelo NIST como K-PKE, e a parte de encapsulamento de chaves, que seria o próprio ML-KEM.

4.1 Esquema de componentes K-PKE

O ML-KEM utiliza como subprocesso um conjunto de algoritmos chamado K-PKE, que consiste nos algoritmos de geração de chaves, cifragem e decifragem. No processo de geração de chaves, são geradas as chaves públicas e privadas. Na etapa de cifragem, ocorre a cifragem de uma mensagem de entrada usando as chaves públicas. Por último, na fase de decifragem, é realizada a decifragem de uma mensagem cifrada usando a chave privada.

Antes de abordar os principais algoritmos do K-PKE, é importante analisar algumas funções que impactam diretamente no seu funcionamento.

$$\begin{aligned} Comprimir_q(x,d) &= \lceil (2^d/q)x \rfloor \ \mathbf{mod}^+ \ 2^d, \\ Descomprimir_q(x,d) &= \lceil (q/2^d)x \rfloor \end{aligned}$$

As funções de comprimir e descomprimir servem para criar uma tolerância de erros durante a cifragem e decifragem de uma mensagem, na qual x é um byte da

mensagem, q é o parâmetro estabelecido do NIST e d para este caso sempre será o valor 1. A função descomprimir, utilizada na cifragem, é aplicada a cada coeficiente do polinômio que representa uma mensagem cifrada, onde os coeficientes são mantidos em zero caso forem zero ou substituídos por $\lceil q/2^d \rfloor$ se forem 1. A função de comprimir é utilizada para fazer o processo reverso, em que se o coeficiente for mais próximo de $\lceil q/2^d \rfloor$ do que de zero, retorna um, caso contrário retorna zero.

Algoritmo 5: Distribuição Binomial Centrada

Entrada: Vetor de bytes $B = b_0, b_1, ..., b_{64\eta-1}$

Saída: Um polinômio $f \in \mathbb{Z}_q[x]/\langle x^n + 1\rangle$

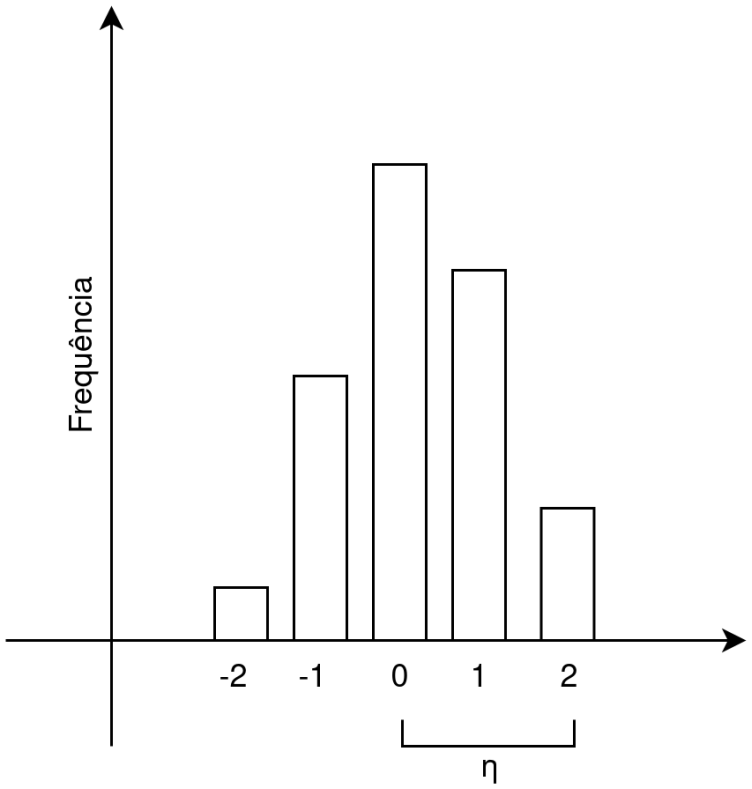
1: **para** $i \leftarrow 0$ *até* $n-1$ **faça**
2: $\quad a \leftarrow_{aleatório} [0, \eta]$
3: $\quad b \leftarrow_{aleatório} [0, \eta]$
4: $\quad f_i \leftarrow a - b$
5: **retorna** $f_0 + f_1 x + f_2 x^2 + ... + f_{n-1} x^{n-1}$

O Algoritmo 5 retorna um polinômio de grau n com os coeficientes baseados na distribuição binomial centrada. A distribuição binomial centrada neste caso é uma distribuição de probabilidade com centro em zero, como ilustra a Figura 4.1. Note que os coeficientes do polinômio resultante do Algoritmo 5 podem assumir apenas os valores no intervalo $[-\eta, \eta]$, dessa forma, este algoritmo é utilizado para gerar vetores de grau pequeno.

Como pode ser visto na Figura 4.2, a geração de chaves consiste em gerar quatro matrizes de polinômios, sendo as matrizes **A** e **t** as chaves públicas, a matriz **s** a chave privada e a matriz **e** apenas uma matriz aleatória de perturbação que pode ser descartada após o final da operação. Perceba que no Algoritmo 6 a ordem das matrizes estão em função do parâmetro k, e seus elementos são polinômios de grau 256 com coeficientes pertencentes ao intervalo $[0, 3329]$. Os elementos da matriz **A** devem ser gerados o mais aleatório possível, enquanto os elementos das matrizes **s** e **e** devem ser gerados por uma distribuição de probabilidade binomial centrada em zero pelo Algoritmo 5. Segundo (STANDARDS; TECHNOLOGY, 2023), a motivação para a escolha desta distribuição foi para dificultar alguns ataques conhecidos a esquemas baseados no LWE. O Algoritmo 6 representa este processo de geração de chaves.

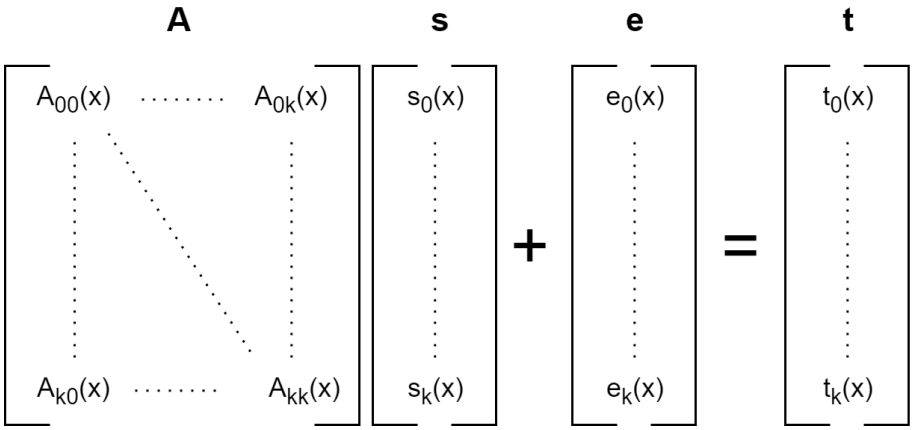
A cifragem de uma mensagem conforme ilustra a Figura 4.3 consiste na geração

Figura 4.1: Distribuição binomial discreta centrada em 0.



Fonte: O autor.

Figura 4.2: Ilustração da geração de chaves do K-PKE no modelo matricial.



Fonte: O autor.

de uma matriz **u** e um polinômio v, isto é, uma mensagem cifrada pelo K-PKE será representada por estes dois elementos. O processo de cifragem de uma mensagem M, representada por um polinômio, consiste em calcular a transposta das chaves públicas **A** e **t**, gerar duas matrizes e um polinômio de forma aleatória que seguem a distribuição binomial centrada em zero, e computar a matriz **u** e o polinômio v como indica o Algoritmo

Algoritmo 6: K-PKE - Geração de chaves

Saída: Chave privada $s \in [\mathbb{Z}_q[x]/\langle x^n+1\rangle]^k$

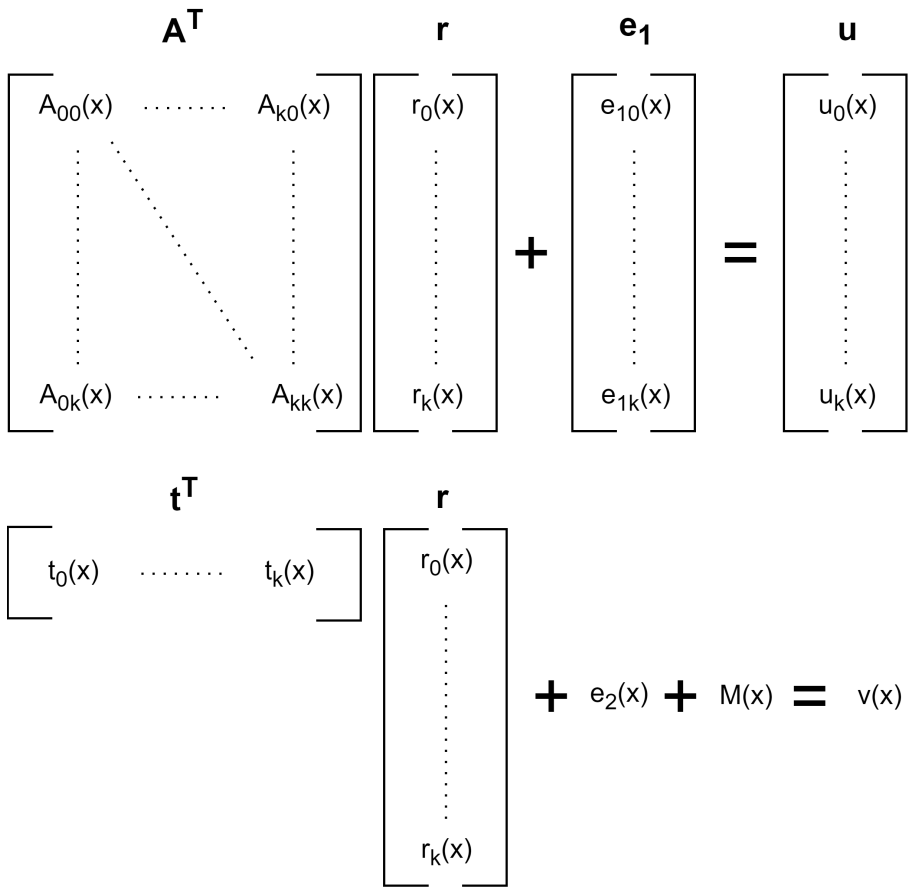
Saída: Chave pública $t \in [\mathbb{Z}_q[x]/\langle x^n+1\rangle]^k$

Saída: Chave pública $A \in [\mathbb{Z}_q[x]/\langle x^n+1\rangle]^{k\times k}$

1: **para** $i \leftarrow 0$ *até* $k-1$ **faça**
2: | **para** $j \leftarrow 0$ *até* $k-1$ **faça**
3: | | A[i][j] $\leftarrow$ random_poly()
4: **para** $i \leftarrow 0$ *até* $k-1$ **faça**
5: | s[i] $\leftarrow$ random_poly_cbd(η_1)
6: **para** $i \leftarrow 0$ *até* $k-1$ **faça**
7: | e[i] $\leftarrow$ random_poly_cbd(η_1)
8: $t \leftarrow As + e$
9: **retorna** *s,t,A*

7.

Figura 4.3: Ilustração da cifragem de uma mensagem usando K-PKE no modelo matricial.



Fonte: O autor.

Algoritmo 7: K-PKE - Cifragem

Entrada: Chave pública A

Entrada: Chave pública t

Entrada: Mensagem m de 32 bytes

Saída: Texto cifrado $u \in [\mathbb{Z}_q[x]/\langle x^n + 1\rangle]^k$

Saída: Texto cifrado $v \in \mathbb{Z}_q[x]/\langle x^n + 1\rangle$

1: $A^T \leftarrow CalculaTransposta(A, k)$

2: $t^T \leftarrow CalculaTransposta(t, k)$

3: **para** $i \leftarrow 0$ *até* $k - 1$ **faça**

4: | r[i] $\leftarrow$ random_poly_cbd(η_1)

5: **para** $i \leftarrow 0$ *até* $k - 1$ **faça**

6: | e_1[i] $\leftarrow$ random_poly_cbd(η_2)

7: $e_2 \leftarrow$ random_poly_cbd(η_2)

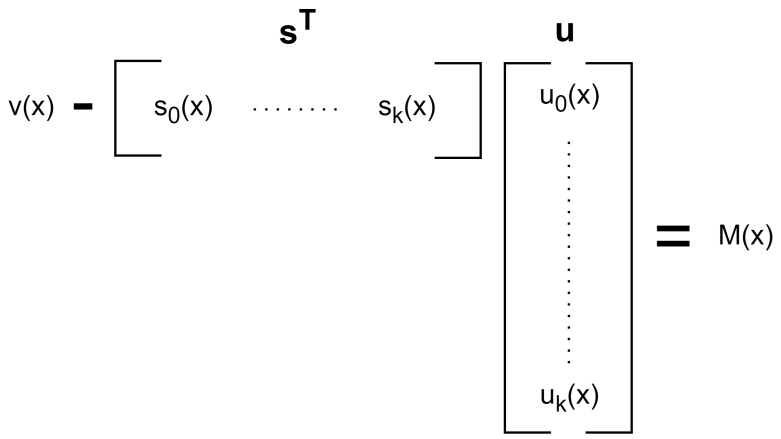
8: $u \leftarrow A^T r + e_1$

9: $v \leftarrow t^T r + e_2 + Descomprimir_q(m, 1)$

10: **retorna** *u,v*

A decifragem do K-PKE é relativamente simples, como pode ser visto na Figura 4.4, basta calcular a transposta da chave privada **s**, computar $m' = v - s^T u$ e aplicar a função de comprimir em m' conforme o Algoritmo 8.

Figura 4.4: Ilustração da decifragem de uma mensagem usando K-PKE no modelo matricial.



Fonte: O autor.

Para entender o porquê deste processo de cifragem e decifragem funcionar,

Algoritmo 8: K-PKE - Decifragem

Entrada: Chave privada s

Entrada: Texto cifrado $u \in [\mathbb{Z}_q[x]/\langle x^n + 1\rangle]^k$

Entrada: Texto cifrado $v \in \mathbb{Z}_q[x]/\langle x^n + 1\rangle$

Saída: Mensagem m de 32 bytes

1: $s^T = CalculaTransposta(s, k)$

2: $m' = v - s^T u$

3: $m = Comprimir_q(m', 1)$

4: **retorna** m

observe a seguinte derivação da equação de decifragem do Algoritmo 8.

$$
\begin{gathered}
v - s^T u \\
(t^T r + e_2 + \lceil q/2 \rfloor M) - (s^T (A^T r + e_1)) \\
((As + e)^T r + e_2 + \lceil q/2 \rfloor M) - (s^T (A^T r + e_1)) \\
((As)^T r + e^T r + e_2 + \lceil q/2 \rfloor M) - (s^T A^T r + s^T e_1) \\
(s^T A^T r + e^T r + e_2 + \lceil q/2 \rfloor M) - (s^T A^T r + s^T e_1) \\
\lceil q/2 \rfloor M + e^T r + e_2 - s^T e_1
\end{gathered}
$$

Perceba que a função Descomprimir multiplica cada byte da mensagem M(x) por $q/2$ no processo de cifragem, e depois são adicionados ruídos de norma pequena. Na última linha da derivação temos uma mensagem M com coeficientes grandes adicionado de ruídos com norma pequena, uma vez que foram gerados seguindo uma distribuição de probabilidade centrada em zero. Desta forma é possível remover os ruídos realizando um arredondamento através da função Comprimir.

Segue um exemplo prático das três etapas do K-PKE com os parâmetros ajustados em $n = 4$, $k = 2$, $q = 3329$, $\eta_1 = 2$ e $\eta_2 = 2$, para simplificar o processo.

Geração de chaves

O primeiro passo é gerar a matriz **A** que será uma das chaves públicas, a matriz **A** neste caso é uma matriz 2×2, então é necessário gerar quatro polinômios de grau 3 com coeficientes entre 0 e 3329. Os coeficientes devem ser o mais aleatório possível.

$$
\mathbf{A} = \begin{bmatrix} 2791 + 1035x + 1916x^2 + 339x^3 & 2154 + 2311x + 517x^2 + 2496x^3 \\ 632 + 3048x + 416x^2 + 930x^3 & 1029 + 699x + 1587x^2 + 524x^3 \end{bmatrix}
$$

Para gerar a matriz **s** e a matriz **e** é necessário gerar uma matriz coluna 2×1 com polinômios de grau 3 e com coeficientes entre -2 e 2. Segundo a distribuição binomial centrada em zero, os polinômios de **s** devem ter mais coeficientes próximos à 0.

$$\mathbf{s} = \begin{bmatrix} -2x + x^2 + x^3 \\ 1 - 1x + -2x^2 - 1x^3 \end{bmatrix} \mathbf{e} = \begin{bmatrix} 1 + 1x + 2x^3 \\ -1 - 1x + 1x^2 - 1x^3 \end{bmatrix}$$

Por fim, para gerar a chave pública **t**, deve-se computar t = **A s** + **e**.

$$\mathbf{t} = \begin{bmatrix} 2394 + 1159x + 105x^2 + 1857x^3 \\ 492 + 3023x + 2948x^2 + 2686x^3 \end{bmatrix}$$

Deve-se ter cuidado ao realizar as operações entre polinômios, pois estes elementos pertencem ao anel quociente $\mathbb{Z}_{3329}[x]/\langle x^4 + 1\rangle$. Um exemplo de como trabalhar com as operações de soma, subtração e multiplicação pode ser conferida no Apêndice A.4 nos Exemplos 17 e 18.

Cifragem

Seja M = 1101 uma mensagem que deseja-se cifrar com K-PKE, representada pelo polinômio M(x) = $1 + 1x + 0x^2 + 1x^3 = 1 + x + x^3$. Primeiro deve-se calcular a transposta das matrizes **A** e **t**.

$$\mathbf{A}^T = \begin{bmatrix} 2791 + 1035x + 1916x^2 + 339x^3 & 632 + 3048x + 416x^2 + 930x^3 \\ 2154 + 2311x + 517x^2 + 2496x^3 & 1029 + 699x + 1587x^2 + 524x^3 \end{bmatrix}$$

$$\mathbf{t}^T = \begin{bmatrix} 2394 + 1159x + 105x^2 + 1857x^3 & 492 + 3023x + 2948x^2 + 2686x^3 \end{bmatrix}$$

O próximo passo é gerar as matrizes de ruído aleatório **r** e e_1, e o polinômio de ruído e_2.

$$\mathbf{r} = \begin{bmatrix} -1 + x + x^2 + 2x^3 \\ 1 + x^3 \end{bmatrix} \mathbf{e}_1 = \begin{bmatrix} 1 + x + x^2 \\ -1 - x + x^2 - x^3 \end{bmatrix} e_2(x) = -x^2 + x^3$$

A cifragem de uma mensagem M(x) gera dois elementos, uma matriz **u** e um polinômio v(x). A geração da matriz **u** é obtida computando $\mathbf{A}^T\mathbf{r} + \mathbf{e}_1$, o resultado desta operação segue abaixo.

$$\mathbf{u} = \begin{bmatrix} 246 + 218x + 719x^2 + 3098x^3 \\ 527 + 2082x + 20x^2 + 2863x^3 \end{bmatrix}$$

Antes de realizar o cálculo de v(x) é preciso aplicar a função Descomprimir a mensagem M(x), que consiste em multiplicar cada coeficiente da mensagem M(x) por $q/2^d$ e arredondar para o inteiro mais próximo. Segundo a documentação do NIST os parâmetros q e d da função Descomprimir são respectivamente 3329 e 1. Desta forma, a mensagem M(x) passa a ser $M(x) = 1665 + 1665x + 1665x^3$. Com isso, para obtermos o último elemento da mensagem cifrada, o polinômio v(x), deve-se computar $\mathbf{t}^T\ \mathbf{r}$ + e_2(x) + M(x).

$$v(x) = 2447 + 908x + 3323x^2 + 2381x^3$$

Decifragem

Para realizar a decifragem de **u** e v(x), é necessário calcular a transposta da chave privada **s** e computar $m'(x) = v - \mathbf{s}^T u$, após computado $m'(x)$ deve-se aplicar a função Comprimir a ela para realizar o arredondamento, assim obtendo a mensagem original M(x).

$$\mathbf{s}^T = \begin{bmatrix} -2x + x^2 + x^3 & 1 - 1x + -2x^2 - 1x^3 \end{bmatrix}$$

$$m'(x) = 2447 + 908x + 3323x^2 + 2381x^3 -$$
$$\begin{bmatrix} -2x + x^2 + x^3 & 1 - 1x + -2x^2 - 1x^3 \end{bmatrix} \begin{bmatrix} 246 + 218x + 719x^2 + 3098x^3 \\ 527 + 2082x + 20x^2 + 2863x^3 \end{bmatrix}$$

$$m'(x) = 1663 + 1245x + 206x^2 + 1874x^3$$

Aplicando a função Comprimir em m'(x), obtemos a mensagem original M, finalizando o processo de decifragem do K-PKE.

$$M(x) = 1 + 1x + 0x^2 + 1x^3$$
$$M = 1101$$

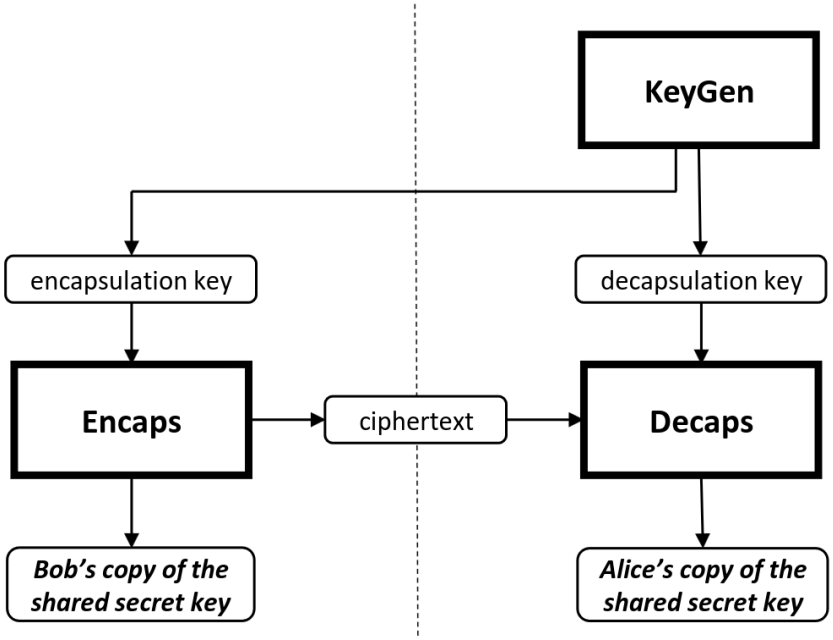
4.2 Encapsulamento de chaves ML-KEM

O ML-KEM é um algoritmo de encapsulamento de chave, ou seja, como dito na Seção 2.4, os algoritmos de encapsulamento de chave têm como objetivo estabelecer uma chave

simétrica aleatória em comum entre duas partes. A principal diferença entre o ML-KEM e o K-PKE, está na mensagem cifrada, enquanto o K-PKE permite cifrar uma mensagem qualquer, o ML-KEM determina a mensagem cifrada, esta mensagem é a chave simétrica compartilhada. A nomenclatura de alguns termos também é diferente, as chaves públicas são chamadas de chaves de encapsulamento, a chave privada é dita de desencapsulamento.

O funcionamento do ML-KEM é semelhante ao K-PKE, entretanto a forma como essa comunicação deve ocorrer é pré-estabelecida, e deve seguir o diagrama da Figura 4.5 segundo o NIST. Perceba que não são cifradas e decifradas mensagens quaisquer como no K-PKE. Em vez disso, o algoritmo gera esta mensagem que é a chave simétrica compartilhada de forma aleatória. A razão disso se dá pelo fato do ML-KEM ser um algoritmo de encapsulamento de chave, este abordado no Capítulo 2.

Figura 4.5: Estabelecimento de uma chave compartilhada com o ML-KEM.



Fonte: (STANDARDS; TECHNOLOGY, 2023).

No modelo simplificado da geração de chaves do ML-KEM, descrito pelo Algoritmo 9, a geração de chaves é idêntica à geração de chaves do K-PKE, com apenas algumas renomeações nas chaves criptográficas, em que dk_pke_s é a chave privada de desencapsulamento, ek_pke_t e ek_pke_a são as chaves públicas de encapsulamento.

O algoritmo de encapsulamento gera uma chave privada aleatória K, que será a chave compartilhada. A chave compartilhada K é cifrada pela chave de encapsulamento

A implementação na linguagem C dos algoritmos 9, 10 e 11 pode ser encon-

Algoritmo 9: ML-KEM - Geração de chaves

Saída: Chave de desencapsulamento $dk\_pke\_s \in [\mathbb{Z}_q[x]/\langle x^n+1\rangle]^k$

Saída: Chave de encapsulamento $ek\_pke\_t \in [\mathbb{Z}_q[x]/\langle x^n+1\rangle]^k$

Saída: Chave de encapsulamento $ek\_pke\_a \in [\mathbb{Z}_q[x]/\langle x^n+1\rangle]^{k\times k}$

1: k_pke_s, k_pke_t, k_pke_a ← k_pke_keygen()

2: dk_pke_s ← k_pke_s

3: ek_pke_t ← k_pke_t

4: ek_pke_a ← k_pke_a

5: **retorna** *dk_pke_s, ek_pke_t, ek_pke_a*

Algoritmo 10: ML-KEM - Encapsulamento

Entrada: Chave de encapsulamento $ek\_pke\_t \in [\mathbb{Z}_q[x]/\langle x^n+1\rangle]^k$

Entrada: Chave de encapsulamento $ek\_pke\_a \in [\mathbb{Z}_q[x]/\langle x^n+1\rangle]^{k\times k}$

Saída: Chave compartilhada cifrada $c\_u \in [\mathbb{Z}_q[x]/\langle x^n+1\rangle]^k$

Saída: Chave compartilhada cifrada $c\_v \in [\mathbb{Z}_q[x]/\langle x^n+1\rangle]^k$

Saída: Chave compartilhada $K \in \mathbb{Z}_q[x]/\langle x^n+1\rangle$

1: K ← random_poly()

2: c_u, c_v ← k_pke_cifragem(ek_pke_a, ek_pke_t, K)

3: **retorna** *c_u, c_v, K*

Algoritmo 11: ML-KEM - Desencapsulamento

Entrada: Chave compartilhada cifrada $c\_u \in [\mathbb{Z}_q[x]/\langle x^n+1\rangle]^k$

Entrada: Chave compartilhada cifrada $c\_v \in [\mathbb{Z}_q[x]/\langle x^n+1\rangle]^k$

Entrada: Chave de desencapsulamento $dk\_pke\_s \in [\mathbb{Z}_q[x]/\langle x^n+1\rangle]^k$

Saída: Chave compartilhada $K \in \mathbb{Z}_q[x]/\langle x^n+1\rangle$

1: K ← k_pke_decifragem(c_u, c_v, dk_pke_s)

2: **retorna** K

trada no repositório do GitHub em https://github.com/opallace/ML-KEM, e também outras ferramentas de teste e *benchmark*. Este repositório conta com uma implementação fiel dos algoritmos deste capítulo, e também com um manual de uso e exemplos básicos de códigos para facilitar o uso por terceiros. É importante lembrar que os algoritmos apresentados neste trabalho e implementados no repositório não devem serem utilizados em ambientes reais. Estes foram desenvolvidos e simplificados para fins acadêmicos.

4.3 Parâmetros

Os parâmetros estabelecidos para o ML-KEM constam na FIPS 203 do NIST em (STANDARDS; TECHNOLOGY, 2023). São fornecidos parâmetros para 3 modelos do ML-KEM com relação ao seu nível de segurança: ML-KEM-512, ML-KEM-768 e ML-KEM-1024. A Tabela 4.1 mostra os parâmetros para estes três modelos.

Tabela 4.1: Parâmetros do ML-KEM.

	n	k	q	η_1	η_2	δ	NIST *Level Security*
ML-KEM-512	256	2	3329	3	2	2^{-139}	I
ML-KEM-768	256	3	3329	2	2	2^{-169}	III
ML-KEM-1024	256	4	3329	2	2	2^{-174}	V

Fonte: O autor.

O parâmetro δ é a probabilidade de erro no processo de decifragem. Os níveis de segurança do NIST vão de I a V, em que os níveis I, III e V significam ser pelo menos tão difícil de quebrar quanto o AES128, AES192 e AES256, respectivamente, através do método de busca exaustiva da chave.

Para alterar estes parâmetros na implementação do GitHub basta modificar os valores definidos no arquivo params.h e recompilar todo o projeto. Os exemplos de geração de chaves, cifragem e decifragem demonstrados na Seção 4.1 utilizaram os parâmetros $n = 4$, $k = 2$, $q = 3329$, $\eta_1 = 2$ e $\eta_2 = 2$ a fim de simplificar os cálculos envolvidos. Entretanto, devido ao parâmetro q estar diretamente relacionado com a taxa de sucesso no processo de decifragem, foi escolhido não modificar.

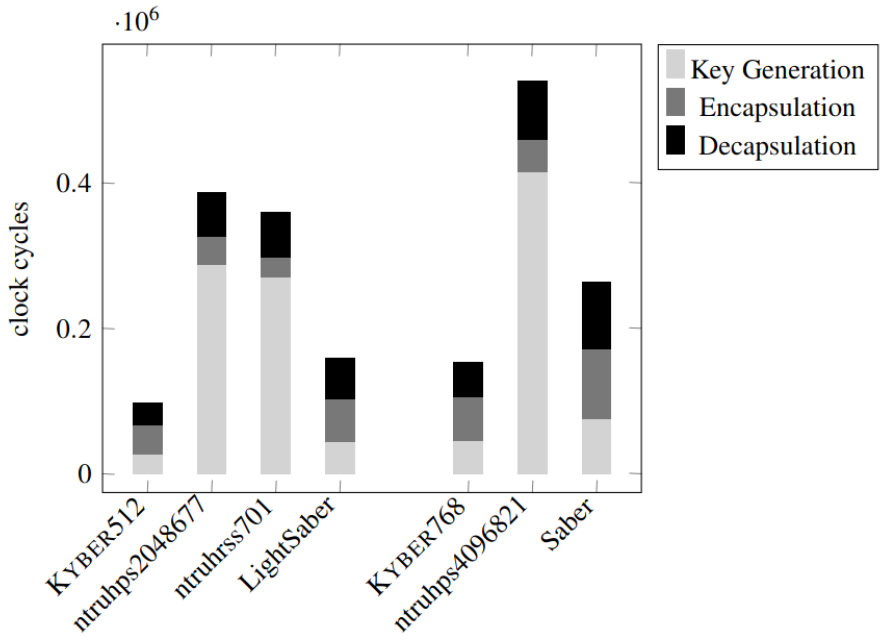
4.4 Segurança

A segurança do ML-KEM está diretamente relacionada com a dificuldade de se resolver o problema MLWE, abordado no Capítulo 3.4. Ou seja, encontrar um algoritmo que fornecidos as chaves públicas $\mathbf{A}$ e $\mathbf{t}$, encontre a chave privada $\mathbf{s}$ com alta probabilidade. Segundo (AVANZI et al., 2012), o ML-KEM é considerado seguro para todos os ataques clássicos e quânticos conhecidos, para os parâmetros estabelecidos na Tabela 4.1.

4.5 Desempenho

O ML-KEM destaca-se em tempo de geração de chaves, encapsulamento e desencapsulamento comparado a outros algoritmos de encapsulamento de chaves do programa NIST PQC, como pode ser visto na Figura 4.6.

Figura 4.6: Comparação de desempenho entre os algoritmos de encapsulamento no programa NIST PQC.



Fonte: Adaptado de (ALAGIC et al., 2022).

Isso se deve ao fato dos seus cálculos necessitarem principalmente de soma e multiplicação de matrizes e polinômios, que são operações de baixa complexidade. A soma de matrizes e polinômios, por exemplo, possui complexidade $\mathcal{O}(n)$ e a multiplicação

matricial $\mathcal{O}(n^2)$. Pelo fato das matrizes serem pequenas, 2×2, 3×3 ou 4×4, dependendo do parâmetro k escolhido, a complexidade do algoritmo em geral é mais afetada pela multiplicação dos elementos dessas matrizes, que no caso são polinômios, e das primitivas criptográficas utilizadas durante os processos (estas primitivas foram removidas deste trabalho para sua simplificação). A multiplicação dos polinômios, pelo fato de pertencerem a um anel quociente, podem ser otimizadas pelo método NTT, possuindo assim complexidade logarítmica ao invés de quadrática. Estes fatores de otimização contribuíram significativamente para seu desempenho, e em sua seleção para padronização.

5 Conclusão

O algoritmo de criptografia ML-KEM é relevante para manter a segurança dos dados em um futuro onde a computação quântica terá capacidade computacional para quebrar a segurança dos principais algoritmos de criptografia atuais. Em 2016, foi publicado o algoritmo Kyber no programa PQC do NIST, com o objetivo de ser o principal algoritmo de criptografia de chave pública, este foi escolhido e padronizado em 2023 e renomeado para ML-KEM. O ML-KEM possui sua segurança baseada na dificuldade de se resolver o problema MLWE (LANGLOIS; STEHLE, 2012). O fato deste problema ser recente implica na carência de materiais de fácil compreensão sobre o assunto, tendo em sua maioria artigos que exigem um alto nível de conhecimento para se compreender. Esta foi uma das dificuldades na elaboração deste trabalho, além da necessidade de conhecimento em diversas áreas como álgebra abstrata, teoria dos números, espaços métricos, álgebra linear e criptologia.

Dentre os objetivos específicos propostos no Capítulo 1, a implementação simplificada do algoritmo ML-KEM, desenvolvida neste trabalho, fornece uma base para que o leitor possa compreender o funcionamento deste algoritmo, e também, a partir deste, possa desenvolver por conta um modelo mais próximo da versão oficial que envolvem conceitos mais sofisticados de segurança e otimização.

Os trabalhos futuros que podem ter como base este documento incluem um aprofundamento nos problemas computacionais baseados em reticulados, além de otimizações utilizadas nas operações do ML-KEM como NTT. Outro conceito recente de criptografia é a criptografia homomórfica, este possui estudos que utilizam variantes do problema LWE.

Referências Bibliográficas

AJTAI, M.; KUMAR, R.; SIVAKUMAR, D. A sieve algorithm for the shortest lattice vector problem. In: . [S.l.]: Association for Computing Machinery, 2001. (STOC '01), p. 601–610.

ALAGIC, G.; APON, D.; COOPER, D. e. a. *Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process*. 2022. Https://csrc.nist.gov/publications/detail/nistir/8413/final. Last accessed 16 September 2022.

ALAGIC, G. et al. *Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process*. NIST Interagency/Internal Report (NISTIR), National Institute of Standards and Technology, Gaithersburg, MD, 2022-07-05 04:07:00 2022. Disponível em: <https://tsapps.nist.gov/publication/get_pdf-.cfm?pub_id=934458>.

ARUTE, F. e. a. Quantum supremacy using a programmable superconducting processor. *Nature*, v. 574, p. 505–510, 2019.

AVANZI, R. et al. *CRYSTALS-Kyber algorithm specifcations and supporting documentation*. 2012. Submission to round 3 of the NIST post-quantum project. https://pq-crystals.org/kyber/data/kyber-specification-round3-20210804.pdf. Disponível em: <https://pq-crystals.org/kyber/data/kyber-specification-round3-20210804.pdf>.

BABAI, L. On lovász' lattice reduction and the nearest lattice point problem (shortened version). In: *Proceedings of the 2nd Symposium of Theoretical Aspects of Computer Science*. Berlin, Heidelberg: Springer-Verlag, 1985. (STACS '85), p. 13–20. ISBN 3540139125.

BARROS, C. F. de. *Autenticação e criptografia pós-quântica baseada em reticulados*. Dissertação (Mestrado) — Universidade Federal do Rio de Janeiro, Instituto de Matemática, Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, 2014.

BELLOVIN, S. M.; BUSH, R. *Security Through Obscurity Considered Dangerous*. [S.l.], mar. 2002. Disponível em: <https://datatracker.ietf.org/doc/draft-ymbk-obscurity/00-/>.

BENNETT, C. H.; BRASSARD, G. Quantum cryptography: Public key distribution and coin tossing. *Theoretical Computer Science*, v. 560, p. 7–11, 2014. ISSN 0304-3975. Theoretical Aspects of Quantum Cryptography – celebrating 30 years of BB84. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0304397514004241>.

BOAS, P. V. E. *Another NP-complete problem and the complexity of computing short vectors in a lattice*. 1981. Technical Report 8104. University of Amsterdam, Department of Mathematics, Netherlands. https://staff.fnwi.uva.nl/p.vanemdeboas/vectors/mi8104c.html. Disponível em: <https://staff.fnwi.uva.nl/p.vanemdeboas/vectors/mi8104c.html>.

CALIFORNIA, B. C. S. D. University of; BACH, E. *Discrete Logarithms and Factoring*. University of California, Computer Science Division (EECS), 1984. (Berkeley UCB/CSD). Disponível em: <https://books.google.com.br/books?id=YlpqGwAACAAJ>.

CHEN, C. et al. *NTRU*. 2019. Submission to round 2 of the NIST post-quantum project. https://ntru.org/f/ntru-20190330.pdf. Disponível em: <https://ntru.org/f/ntru-20190330-.pdf>.

DELFS, H.; KNEBL, H. *Introduction to Cryptography*. 2. ed. [S.l.]: springer, 2007.

DIFFIE, W.; HELLMAN, M. New directions in cryptography. *IEEE Transactions on Information Theory*, v. 22, n. 6, p. 644–654, 1976.

DOOLEY, J. F. *History of Cryptography and Cryptanalysis*. [S.l.]: Springer, 2018.

DUCAS, L. et al. *CRYSTALS-Dilithium (Version 3.1)*. 2021. Submission to round 3 of the NIST post-quantum project. https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf. Disponível em: <https://pq-crystals.org-/dilithium/data/dilithium-specification-round3-20210208.pdf>.

DWORKIN, M. et al. *Advanced Encryption Standard (AES)*. [S.l.]: Federal Inf. Process. Stds. (NIST FIPS), National Institute of Standards and Technology, Gaithersburg, MD, 2001-11-26 2001.

FERNANDES, R. L.; RICOU, M. *Introdução à Álgebra*. [S.l.]: IST Press, 2004.

FINCKE, U.; POHST, M. Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. *Mathematics of Computation*, v. 44, p. 463–471, 1985.

FOUQUE, P.-A. et al. *Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU*. 2020. Submission to round 3 of the NIST post-quantum project. https://falcon-sign.info/falcon.pdf. Disponível em: <https://falcon-sign.info/falcon.pdf>.

GAMBETTA, J. *Quantum-centric supercomputing: The next wave of computing*. 2022. Https://research.ibm.com/blog/next-wave-quantum-centric-supercomputing. Last accessed 28 Match 2023.

GOLDREICH, O. et al. Approximating shortest lattice vectors is not harder than approximating closest lattice vectors. *Information Processing Letters*, v. 71, n. 2, p. 55–61, 1999. ISSN 0020-0190.

GROVER, L. K. A fast quantum mechanical algorithm for database search. In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*. Association for Computing Machinery, 1996. (STOC '96), p. 212–219. Disponível em: <https://doi.org/10.1145/237814.237866>.

HAMDI, S. M. et al. A compare between shor's quantum factoring algorithm and general number field sieve. In: *2014 International Conference on Electrical Engineering and Information & Communication Technology*. [S.l.: s.n.], 2014. p. 1–6.

KANNAN, R. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research*, v. 12, p. 415–440, 1987.

KATZ, J.; LINDELL, Y. *Introduction To Modern Cryptography*. 3. ed. [S.l.]: CRC PRESS, 2021.

KERCKHOFFS, A. La cryptographie militaire. *Journal des sciences militaires*, v. 9, p. 5–38, 1883.

LANGLOIS, A.; STEHLE, D. *Worst-Case to Average-Case Reductions for Module Lattices*. 2012. Cryptology ePrint Archive, Paper 2012/090. https://eprint.iacr.org/2012/090. Disponível em: <https://eprint.iacr.org/2012/090>.

LENSTRA, A. K.; LENSTRA, H. W.; LOVÁSZ, L. Factoring polynomials with rational coefficients. *Mathematische Annalen*, v. 261, p. 515—534, 1982.

LYUBASHEVSKY, V.; PEIKERT, C.; REGEV, O. On ideal lattices and learning with errors over rings. In: GILBERT, H. (Ed.). *Advances in Cryptology – EUROCRYPT 2010*. [S.l.]: Springer Berlin Heidelberg, 2010. p. 1–23. ISBN 978-3-642-13190-5.

MICCIANCIO, D.; GOLDWASSER, S. *Complexity of Lattice Problems: a cryptographic perspective*. Boston, Massachusetts: Kluwer Academic Publishers, 2002. (The Kluwer International Series in Engineering and Computer Science, v. 671).

MOODY, D. *Round 2 of the NIST PQC "Competition- What was NIST Thinking?* 2019. PQCrypto 2019 in Chongqing, China. https://csrc.nist.gov/Presentations/2019/Round-2-of-the-NIST-PQC-Competition-What-was-NIST. Disponível em: <https://csrc.nist-.gov/Presentations/2019/Round-2-of-the-NIST-PQC-Competition-What-was-NIST>.

NIST. *Post-Quantum Cryptography Standardization*. 2017. Https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization. Last accessed 16 September 2022.

REGEV, O. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, Association for Computing Machinery, v. 56, n. 6, p. 84—93, 2009.

RIVEST, R. L.; SHAMIR, A.; ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, Association for Computing Machinery, New York, NY, USA, v. 21, n. 2, p. 120–126, feb 1978. ISSN 0001-0782. Disponível em: <https://doi.org/10.1145/359340.359342>.

SALARIFARD, R.; SOLEIMANY, H. *Efficient Accelerator for NTT-based Polynomial Multiplication*. 2023. Cryptology ePrint Archive, Paper 2023/686. https://eprint.iacr.org/2023/686. Disponível em: <https://eprint.iacr.org/2023/686>.

SCHNEIER, B. *Applied Cryptography*. [S.l.]: John Wiley & Sons, Inc., 1996.

SHANNON, C. E. *Mathematical Theory of Cryptography*. 1945. Bell System Technical Memo. Disponível em: <https://www.iacr.org/museum/shannon/shannon45.pdf>.

SHOR, P. W. Algorithms for quantum computation: discrete logarithms and factoring. In: *Proceedings 35th Annual Symposium on Foundations of Computer Science*. [S.l.: s.n.], 1994. p. 124–134.

STALLINGS, W. *Criptografia e seguranca de redes*. 4. ed. [S.l.]: Pearson Prentice Hall, 2008.

STANDARDS, N. B. of. Data encryption standard. In: . [s.n.], 1977. Disponível em: <https://csrc.nist.gov/CSRC/media/Publications/fips/46/archive/1977-01-15-/documents/NBS.FIPS.46.pdf>.

STANDARDS, N. I. of; TECHNOLOGY. *Module-Lattice-Based Key-Encapsulation Mechanism Standard*. 2023. Initial Public Draft of FIPS 203 in NIST post-quantum project. https://doi.org/10.6028/NIST.FIPS.203.ipd. Disponível em: <https://doi.org/10-.6028/NIST.FIPS.203.ipd>.

YAN, B. et al. *Factoring integers with sublinear resources on a superconducting quantum processor*. 2022.

YUVAL, G. How to swindle rabin. *CRYPTOLOGIA*, v. 3, p. 187–191, 07 1979.

A Conceitos matemáticos

Neste capítulo iremos abordar alguns conceitos matemáticos importantes para o entendimento deste trabalho.

A.1 Aritmética modular

A aritmética modular é um dos conceitos estudados pelo ramo da matemática conhecido como teoria dos números, que será amplamente utilizado neste trabalho. O conhecimento de aritmética modular será necessário para compreender alguns problemas de criptografia abordados nos Capítulo 2 e 3.4. O algoritmo ML-KEM também faz o uso da aritmética modular nas operações aritméticas entre polinômios, na qual é vista com mais detalhes no Apêndice A.4.

Definição 15 (divisibilidade)**.** *Sejam a e b inteiros diferentes de 0, é dito que a é divisível por b se e somente se existe um inteiro c tal que $a = b.c$, a notação usada para denotar essa divisibilidade de a por b é $a \mid b$.*

Teorema A.1.1. *Se $a \mid b$ e $a \mid c$, então $(b + c) \mid a$.*

Teorema A.1.2. *Se $a \mid b$, então $(a.c) \mid b$.*

Teorema A.1.3. *Se $a \mid b$ e $b \mid c$, então $a \mid c$.*

Teorema A.1.4. *Seja $n \in \mathbb{Z}$ e $d \in \mathbb{Z}_+$, existe um único par de inteiros q e r com $0 \leq r < d$ tais que $n = d.q + r$.*

Sejam $n, d, q, r \in \mathbb{Z}$ tais que $d > 0$, $n = d.q + r$ e $0 \leq r < d$, definem-se as funções **div** e **mod** tais que:

- n **div** $d = q$;
- n **mod** $d = r$.

Definição 16 (Congruência modular)**.** *Dados $j, k \in \mathbb{Z}$ e $m \in \mathbb{Z}_+$, é dito que j é congruente a k no módulo m se e somente se $(j - k) \mid m$. A notação de congruência é $j \equiv k$ ($\boldsymbol{mod}$ m), ou seja, j e k são o mesmo número no módulo m.*

Propriedades da congruência modular:

- **Propriedade reflexiva:** Seja $m \in \mathbb{Z}_+$, $n \equiv n \,(\mathbf{mod}\, m)\; \forall n \in \mathbb{Z}$;
- **Propriedade simétrica:** Seja $m \in \mathbb{Z}_+$, $j \equiv k \,(\mathbf{mod}\, m)$ se e somente se $k \equiv j \,(\mathbf{mod}\, m)$;
- **Propriedade transitiva:** Seja $m \in \mathbb{Z}_+$, se $j \equiv k \,(\mathbf{mod}\, m)$ e $k \equiv i \,(\mathbf{mod}\, m)$ então $j \equiv i \,(\mathbf{mod}\, m)$.

Exemplo 1: para facilitar o entendimento sobre conceito de congruência, considere um relógio de 12 horas. Se agora são 5 horas, que horas serão daqui a 10 horas? Como este relógio não possui 15 horas, reinicia-se a contagem a partir de 0 quando o relógio chega em 12 horas. Dessa forma tem-se que $15 = 12 + 3$; neste caso, se agora são 5 horas, daqui a 10 horas serão 3 horas. Desta forma é dito que $15 \equiv 3$ (**mod** 12), visto que $(15 - 3) \mid 12$. ■

A.2 Teoria dos números

Esta seção apresenta alguns conceitos estudados no campo da teoria dos números mencionados na Seção 2.5, onde é abordado os problemas que os algoritmos de criptografia atuais são baseados.

Definição 17 (número primo)**.** *Um número natural maior que 1 que possui como divisores positivos apenas 1 e ele próprio.*

Definição 18 (coprimos)**.** *Dois números são coprimos, ou primos entre si, se o único divisor comum entre eles for o número 1.*

Exemplo 2: os divisores do número 4 são: 2 e 1, e os divisores do número 9 são: 3 e 1. Portando os números 4 e 9 são coprimos. ■

Definição 19 (número composto)**.** *Dado* $n, n_1, n_2 \in \mathbb{N}$ *tal que* $n > 1$, $n \mid n_1$ *e* $n \mid n_2$. *Temos que* n *é dito composto se* $n = n_1 n_2$, *com* $1 < n_1 < n$ *e* $1 < n_1 < n$.

Teorema A.2.1 (teorema fundamental da aritmética)**.** *Todo número natural maior que 1 ou é primo, ou pode ser escrito como um produto de números primos.*

Segundo o teorema fundamental da aritmética dado um número composto, por exemplo o número 145, podemos representá-lo pela multiplicação de números primos, neste caso 5 e 29. O problema da fatoração prima, abordado na Seção 2.5.1 utiliza este teorema.

Definição 20 (ordem multiplicativa)**.** *Dado os números inteiros positivos* n *e* m, *tais que* n *e* m *são coprimos, a ordem multiplicativa de* n *módulo* m *é o menor valor inteiro positivo* k *que satisfaz* $n^k \equiv 1$ (***mod*** m).

Exemplo 3: tem-se que $2^6 \equiv 1$ (**mod** 7). No entanto, $2^3 \equiv 1$ (**mod** 7). Observa-se que ambos satisfazem a congruência; porém, 3 é o menor expoente que satisfaz a equação. Portanto, conclui-se que 3 é a ordem multiplicativa de 2 módulo 7. ■

Definição 21 (função totiente de Euler)**.** *A função totiente de Euler de um número natural* n, *denotada por* $\varphi(n)$, *é a quantidade de números coprimos a* n *no intervalo* $1 \leq \varphi(n) < n$.

Exemplo 4: tem-se $\varphi(10) = 4$ pois a quantidade de números coprimos a 10 no intervalo $[1, 10[$ é o conjunto $\{1, 3, 7, 9\}$ contendo 4 elementos. ■

Definição 22 (raiz primitiva módulo m)**.** *Se a ordem multiplicativa de k módulo m é igual à função totiente de Euler para m, dizemos que k é uma raiz primitiva módulo m.*

Exemplo 5: é dito que 3 é uma raiz primitiva módulo 10, visto que a ordem multiplicativa de 3 e 10 é 4, e como calculado anteriormente, $\varphi(10) = 4$. Sendo assim, como a ordem multiplicativa entre 3 e 10 é igual à função totiente de 10, o número 3 é uma raiz primitiva módulo 10. Este conceito é utilizado no problema do logaritmo discreto, abordado na Seção 2.5.2. ■

A.3 Álgebra linear

Esta seção se dedica a alguns conceitos de álgebra linear essenciais para a compreensão deste trabalho. A álgebra linear é um ramo da matemática que se dedica no estudo de equações lineares, vetores, espaços vetoriais, matrizes, entre outros.

Definição 23 (matriz)**.** *Chamamos de matriz $m \times n$ uma tabela de elementos dispostos em m linhas e n colunas.*

$$\mathbf{A}_{m\times n} = \begin{bmatrix} a_{11} & a_{12} & . & . & . & b_{1n} \\ a_{21} & a_{22} & . & . & . & b_{2n} \\ a_{31} & a_{32} & . & . & . & b_{3n} \\ . & . & . & & & . \\ . & . & & . & & . \\ . & . & & & . & . \\ a_{m1} & a_{m2} & & & & a_{mn} \end{bmatrix} = [a_{ij}]_{m\times n}$$

O número de linhas e colunas é dito a ordem de uma matriz. É importante ressaltar algumas matrizes especiais, como a matriz quadrada, em que o número de linhas é igual o número de colunas; matriz nula, em que todos os elementos são zero, tal matriz é representada por $\mathbf{0}$; matriz coluna, em que se tem apenas uma coluna e analogamente matriz linha, em que se tem apenas uma linha; por fim, matriz identidade quadrada, denotada por $\mathbf{I}$ tal que $a_{ii} = 1$ e $a_{ij} = 0$ para $i \neq j$.

A seguir, é definida as operações e propriedades de soma entre matrizes de mesma ordem, multiplicação de uma matriz por um escalar, operação de transposição e multiplicação entre matrizes.

A adição de matrizes de mesma ordem, isto é $\mathbf{A}_{m\times n} + \mathbf{B}_{m\times n}$ é representada por:

$$\mathbf{A} + \mathbf{B} = [a_{ij} + b_{ij}]_{m\times n}$$

e apresenta as seguintes propriedades:

(i)$\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A}$,

(ii)$\mathbf{A} + (\mathbf{B} + \mathbf{C}) = (\mathbf{A} + \mathbf{B}) + \mathbf{C}$,

(iii)$\mathbf{A} + \mathbf{0} = \mathbf{A}$, onde $\mathbf{0}$ representa matriz nula.

A multiplicação de um escalar k por uma matriz $\mathbf{A} = [a_{ij}]_{m\times n}$ é representada por:

$$k.\mathbf{A} = [ka_{ij}]_{m\times n}$$

e segue as seguintes propriedades, dadas duas matrizes $\mathbf{A}$ e $\mathbf{B}$ de mesma ordem e k, k_1 e k_2 escalares:

(i)$k(\mathbf{A} + \mathbf{B}) = k\mathbf{A} + k\mathbf{B}$,

(ii)$(k_1 + k_2)\mathbf{A} = k_1\mathbf{A} + k_2\mathbf{A}$,

(iii)$0.\mathbf{A} = \mathbf{0}$, neste caso, se multiplicarmos o escalar zero, obtemos a matriz nula,

(iv)$k_1(k_2\mathbf{A}) = (k_1k_2)\mathbf{A}$.

Outra operação importante é a de transposição, seja a matriz $\mathbf{A} = [a_{ij}]_{m\times n}$, sua transposta, denotada $\mathbf{A}^T = [b_{ij}]_{n\times m}$, é tal que $b_{ij} = a_{ji}$.

$$\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 0 & 3 \\ -1 & 4 \end{bmatrix}_{3\times 2} \qquad \mathbf{A}^T = \begin{bmatrix} 2 & 0 & -1 \\ 1 & 3 & 4 \end{bmatrix}_{2\times 3}$$

A operação de transposição respeita as seguintes propriedades:

(i)$(\mathbf{A}^T)^T = \mathbf{A}$,

(ii)$(\mathbf{A} + \mathbf{B})^T = \mathbf{A}^T + \mathbf{B}^T$,

(iii)$(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$,

(iv)$(k\mathbf{A})^T = k\mathbf{A}^T$, onde k é um escalar.

A operação de multiplicação entre matrizes é um pouco mais complexa e só pode ser efetuada se o número de colunas da primeira matriz for igual ao número de linhas da segunda, isto é, sejam as matrizes $\mathbf{A} = [a_{ij}]_{m\times n}$ e $\mathbf{B} = [b_{rs}]_{n\times p}$, definimos a operação $\mathbf{AB} = [c_{uv}]_{m\times p}$, onde

$$c_{uv} = \sum_{k=1}^{n} a_{uk} b_{kv}$$

e respeita as seguintes propriedades:

(i)$\mathbf{AB} \neq \mathbf{BA}$,

(ii)$\mathbf{AI} = \mathbf{IA} = \mathbf{A}$,

(iii)$\mathbf{A}(\mathbf{B} + \mathbf{C}) = \mathbf{AB} + \mathbf{AC}$,

(iv)$(\mathbf{A} + \mathbf{B})\mathbf{C} = \mathbf{AC} + \mathbf{BC}$,

(v)$(\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC})$,

(vi)$(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$,

(vii)$\mathbf{0}.\mathbf{A} = \mathbf{0}$ e $\mathbf{A}.\mathbf{0} = \mathbf{0}$, e $\mathbf{0}$ representa a matriz nula.

Definição 24 (vetores)**.** *Dado um segmento de reta orientado AB, um vetor é o conjunto de todos os segmentos orientados equipolentes*[1] *a AB.*

Seja $\vec{v}$ este conjunto, podemos escrever que $\vec{v} = \{XY \mid XY \sim AB\}$, onde XY é um segmento qualquer do conjunto. O vetor representante desse conjunto inicia na origem.

Definição 25 (vetor nulo)**.** *Dizemos que um vetor é nulo quando sua origem e a extremidade coincidem. Denotamos esse vetor por* $\vec{0}$.

Definição 26 (espaços vetoriais)**.** *Um espaço vetorial é um conjunto* V *de vetores, não vazio, com duas operações: soma entre dois vetores* $\vec{u}, \vec{v} \in V$ *que resulta em* $\vec{v} + \vec{u} \in V$*, e multiplicação entre um vetor* $\vec{v} \in V$ *por escalar* $a \in \mathbb{R}$*, cujo resultado é* $a\vec{v} \in V$*. Tais que para quaisquer* $\vec{u}, \vec{v}, \vec{w} \in V$ *e* $a, b \in \mathbb{R}$*, os seguintes axiomas são satisfeitos:*

Adição*:*

(i)$(\vec{u} + \vec{v}) + \vec{w} = \vec{u} + (\vec{v} + \vec{w})$,

(ii)$\vec{u} + \vec{v} = \vec{v} + \vec{u}$,

[1]Dois segmentos são equipolentes quanto têm a mesma direção, o mesmo sentido e o mesmo comprimento. A relação de equipolência é denotada por $\sim$

(iii)Existe $\vec{0} \in V$ *tal que* $\vec{u} + \vec{0} = \vec{u}$ *e*

(iv)Existe $-\vec{u} \in V$ *tal que* $\vec{u} + (-\vec{u}) = \vec{0}$.

***Multiplicação por escalar**:*

(i)$a(\vec{u} + \vec{v}) = a\vec{u} + a\vec{v}$,

(ii)$(a + b)\vec{v} = a\vec{v} + b\vec{v}$,

(iii)$(ab)\vec{v} = a(b\vec{v})$ *e*

(iv)$1\vec{u} = \vec{u}$.

Exemplo 6: Seja um espaço vetorial o conjunto de vetores $V = \mathbb{R}^n = \{(x_1, x_2, ..., x_n) \mid x_i \in \mathbb{R}\}$. ■

Exemplo 7: $V = M(m, n)$, o conjunto das matrizes reais $m \times n$. ■

Exemplo 8: Tem-se também o espaço vetorial $V = P_n = \{a_0 + a_1x + a_2x^2 + ... + a_nx^n \mid a_i \in \mathbb{R}\}$ definido por todos os polinômios com coeficientes reais de grau menor ou igual a n. ■

Os elementos dos conjuntos dos exemplos 6,7 e 8 respeitam todas as propriedades de um espaço vetorial, portanto são espaços vetoriais.

Definição 27 (produto interno)**.** *Seja* V *um espaço vetorial sobre* $\mathbb{R}$*. O produto interno é definido por* $\langle ., . \rangle : V \times V \to \mathbb{R}$*, onde tal operação satisfaz as seguintes propriedades:*

(i)$\langle \vec{v}, \vec{v} \rangle \geq 0$ *para todo vetor* $\vec{v}$*, e*
$\langle \vec{v}, \vec{v} \rangle = 0$ *se, e somente se* $\vec{v} = \vec{0}$*;*

(ii)$\langle a\vec{v}_1, \vec{v}_2 \rangle = a\langle \vec{v}_1, \vec{v}_2 \rangle$ *para todo* $a \in \mathbb{R}$*;*

(iii)$\langle \vec{v}_1 + \vec{v}_2, \vec{v}_3 \rangle = \langle \vec{v}_1, \vec{v}_3 \rangle + \langle \vec{v}_2, \vec{v}_3 \rangle$*;*

(iv)$\langle \vec{v}_1, \vec{v}_2 \rangle = \langle \vec{v}_2, \vec{v}_1 \rangle$.

Sejam os vetores $\vec{a} = (a_1, a_2, a_3, ..., a_n)$ e $\vec{b} = (b_1, b_2, b_3, ..., b_n)$, o produto interno entre $\vec{a}$ e $\vec{b}$ é dado por $\langle \vec{a}, \vec{b} \rangle = a_1b_1 + a_2b_2 + a_3b_3 + ... + a_nb_n$.

Em um espaço vetorial com produto interno definido, podemos realizar outras operações entre vetores como calcular a norma vetorial, ângulo e a projeção entre dois vetores.

Definição 28 (ortogonalidade entre vetores)**.** *Dois vetores $\vec{u}$, $\vec{v}$ são ditos ortogonais se o ângulo entre eles for de $90°$, em outras palavras, quando o produto interno $\langle \vec{u}, \vec{v} \rangle = 0$, denota-se tal ortogonalidade por $\vec{u} \perp \vec{v}$.*

Definição 29 (norma vetorial)**.** *Norma consiste em uma função que associa um vetor do espaço vetorial em um número real não negativo. A norma de um vetor está associada ao seu comprimento, também denominada como módulo de um vetor.*

Neste trabalho iremos fazer o uso apenas da norma euclidiana. Seja o vetor $\vec{v} = (a_1, a_2, ..., a_n)$, sua norma euclidiana é definida por:

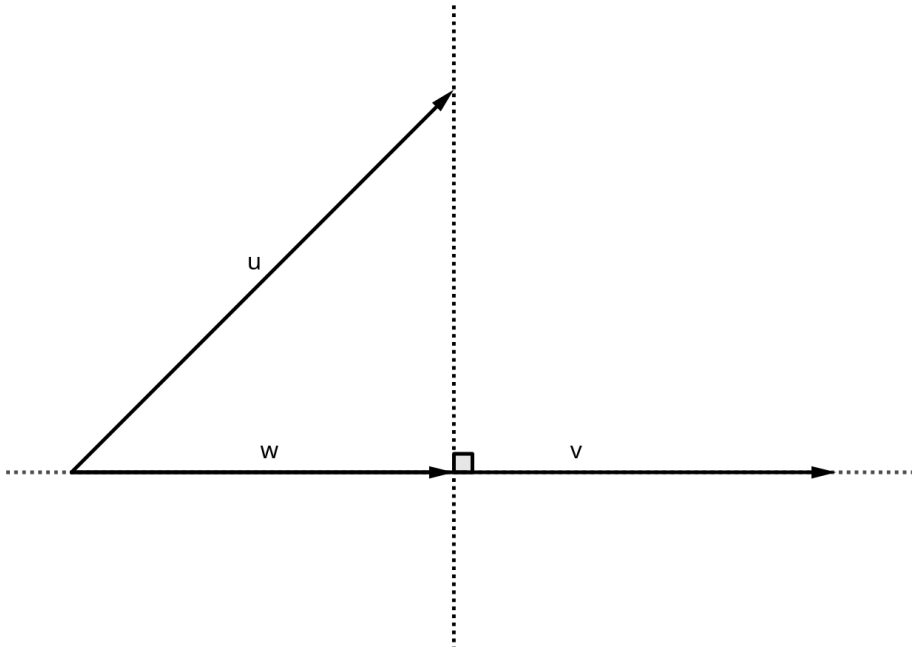
$$\|\vec{v}\| = \sqrt{a_1^2 + a_2^2 + ... + a_n^2}$$

Definição 30 (projeção entre vetores)**.** *A projeção vetorial de um vetor $\vec{u}$ sobre um vetor não nulo $\vec{v}$, denotado por $proj_{\vec{v}}\vec{u}$, é uma projeção ortogonal de $\vec{u}$ sobre uma reta paralela à $\vec{v}$.*

Sejam os vetores $\vec{u}$ e $\vec{v}$, para calcular o vetor projeção $\vec{w}$ de $\vec{u}$ sobre $\vec{v}$, como mostra a Figura A.1, utiliza-se a fórmula:

$$\vec{w} = proj_{\vec{v}}\vec{u} = \frac{\langle \vec{u}, \vec{v} \rangle}{\|\vec{v}\|^2}\vec{v}$$

Figura A.1: Projeção de $\vec{u}$ sobre $\vec{v}$.



Fonte: O autor.

Definição 31 (combinação linear)**.** *Seja V um espaço vetorial, dizemos que um vetor $\vec{v}$ é uma combinação linear de outros vetores $\vec{v_1}, \vec{v_2}, \vec{v_3}, ..., \vec{v_n} \in V$ se e somente se existem escalares $a_1, a_2, a_3, ..., a_n \in \mathbb{R}$ tais que $\vec{v} = a_1\vec{v_1} + a_2\vec{v_2} + a_3\vec{v_3} + ... + a_n\vec{v_n}$.*

Exemplo 9: O vetor $\vec{w} = (5, 3) \in \mathbb{R}^2$, pode ser escrito como uma combinação linear dos vetores $\vec{u} = (1, 0)$ e $\vec{v} = (0, 1)$. Isso pode ser visto por $(5, 3) = 5(1, 0) + 3(0, 1)$. ■

Definição 32 (dependência e independência linear)**.** *Seja V um espaço vetorial qualquer e $\vec{v_1}, \vec{v_2}, \vec{v_3}, ..., \vec{v_n} \in V$, dizemos que o conjunto $\{\vec{v_1}, \vec{v_2}, \vec{v_3}, ..., \vec{v_n}\}$ é linearmente independente quando a combinação linear nula $\vec{0} = a_1\vec{v_1} + a_2\vec{v_2} + a_3\vec{v_3} + ... + a_n\vec{v_n}$ implicar que obrigatoriamente $a_1 = a_2 = a_3 = ... = a_n = 0$. Caso exista algum escalar $a_n \neq 0$ que satisfaça a combinação linear nula, dizemos que o conjunto $\{\vec{v_1}, \vec{v_2}, \vec{v_3}, ..., \vec{v_n}\}$ é linearmente dependente.*

Exemplo 10: os vetores $\vec{u} = (1, 0)$ e $\vec{u} = (0, 1)$ são linearmente independentes pois

$$\begin{aligned} a_1(1,0) + a_2(0,1) &= (0,0) \\ (a_1,0) + (0,a_2) &= (0,0) \\ (a_1,a_2) &= (0,0) \\ a_1 &= 0 \\ a_2 &= 0 \end{aligned}$$

■

Exemplo 11: Seja os vetores $\vec{u} = (1, 2)$ e $\vec{u} = (3, 6)$ do espaço vetorial $\mathbb{R}^2$, tais vetores não são linearmente independentes pois,

$$\begin{aligned} a_1(1,2) + a_2(3,6) &= (0,0) \\ (a_1,2a_1) + (3a_2,6a_2) &= (0,0) \\ (a_1+3a_2, 2a_1+6a_2) &= (0,0) \end{aligned}$$

$$\begin{cases} a_1 + 3a_2 &= 0 \\ 2a_1 + 6a_2 &= 0 \end{cases}$$

resolvendo o sistema acima, encontramos infinitas soluções em que a_1 e a_2 são diferentes de 0, determinando assim que os vetores $\vec{u} = (1, 2)$ e $\vec{u} = (3, 6)$ são linearmente dependentes.

■

Definição 33 (base de espaços vetoriais)**.** *Um conjunto de vetores* $\beta = \{\vec{v_1}, \vec{v_2}, \vec{v_3}, ..., \vec{v_n}\}$ *é considerado uma base para um espaço vetorial* V*, se as seguintes condições forem satisfeitas:*

(i)β *é linearmente independente;*

(ii)para todo vetor $\vec{v} \in V$ *existem* $a_1, a_2, \ldots, a_n \in \mathbb{R}$ *tal que* $\vec{v} = a_1\vec{v_1} + a_2\vec{v_2} + \ldots + a_n\vec{v_n}$.

Definição 34 (dimensão de espaços vetoriais)**.** *A dimensão de um espaço vetorial* V *é a quantidade de elementos de sua base geradora, denotada por* $Dim(V)$.

A.4 Álgebra abstrata

Uma operação binária interna é uma função (eventualmente) parcial $\oplus : \mathbb{A}^2 \to \mathbb{A}$ e é usualmente denotada como um par ordenado ¡$\mathbb{A}, \oplus$¿, no qual $\mathbb{A}$ é denominado conjunto suporte. A seguir são apresentadas algumas propriedades de operações binárias internas.

(i)**fechada:** $\oplus : \mathbb{A}^2 \to \mathbb{A}$ é função total

(ii)**comutativa:** $\forall a, b \in \mathbb{A}\ (a \oplus b = b \oplus a)$

(iii)**associativa:** $\forall a, b \in \mathbb{A}\ ((a \oplus b) \oplus c = (c \oplus b) \oplus a = a \oplus b \oplus c)$

(iv)**elemento neutro:** $\exists e \in \mathbb{A}\ \forall a \in \mathbb{A}\ (a \oplus e = a = e \oplus a)$

(v)**elemento inverso:** $\forall a \in \mathbb{A}\ \exists \bar{a} \in \mathbb{A}\ (a \oplus \bar{a} = e = \bar{a} \oplus a)$

Definição 35 (grupoide)**.** *Um grupoide é um conjunto não vazio junto a uma operação binária fechada. Se um grupoide respeitar a propriedade de comutatividade, denomina-se grupoide abeliano.*

Exemplo 12: a realização da operação de subtração entre os elementos do conjunto dos inteiros é sempre um inteiro, portanto $(\mathbb{Z}, -)$ denota um grupoide não abeliano, visto que se $a, b \in \mathbb{Z}\ a - b \neq b - a$. ■

Definição 36 (semigrupo)**.** *Um semigrupo é definido como um conjunto não vazio que possui uma operação binária fechada e associativa. Se a operação binária também respeitar a propriedade de comutatividade, é dito que o semigrupo é abeliano.*

Definição 37 (monoide)**.** *Os monoides são definidos por um conjunto não vazio e uma operação binária que respeita as propriedades de fechamento, associatividade e possui elemento neutro. Se um monoide também possuir a propriedade de comutatividade, é dito que o monoide é abeliano.*

Exemplo 13: A realização da operação de multiplicação entre os elementos do conjunto $\mathbb{Z}^*$ denotam um semigrupo, visto que respeitam a associatividade, portanto $(\mathbb{Z}^*, *)$ é um semigrupo. Entretanto, a estrutura $(\mathbb{Z}^*, *)$ também pode ser classificada como um monoide, pois possui elemento neutro 1 para a operação de multiplicação. ■

Definição 38 (grupo)**.** *Grupos são definidos como um conjunto não vazio que possui operação binária que respeita as propriedades de fechamento, associatividade, possui elemento neutro e elemento inverso. Se um grupo respeitar a propriedade de comutatividade, é dito que o grupo é abeliano.*

Exemplo 14: A estrutura $(\mathbb{Z}, +)$ é um grupo, visto que a operação de soma entre os elementos de $\mathbb{Z}$ possui as propriedades de fechamento, associatividade, possui elemento neutro 0 e possui elemento inverso $a, a' \in \mathbb{Z}\ a + (-a) = 0$. A estrutura $(\mathbb{Z}, +)$ também possui a propriedade comutativa, sendo assim, classificada como um grupo abeliano. ■

Definição 39 (anel)**.** *Um anel $(A, \oplus, \otimes)$ é uma álgebra constituída de um conjunto suporte e duas operações $\oplus$ e $\otimes$ na qual:*

- *$(A, \oplus)$ é um grupo abeliano;*
- *$\otimes$ é associativo; e*
- *$\forall a, b, c \in A(a \otimes (b \oplus c)) = ((a \otimes b) \oplus (a \otimes c))$.*

Exemplo 15: A estrutura $(\mathbb{Z}, +, *)$ é um anel, pois demonstramos no exemplo 14 que a estrutura $(\mathbb{Z}, +)$ é um grupo abeliano, a operação de multiplicação é associativa entre os elementos de $\mathbb{Z}$ e também respeita a distributividade da multiplicação na soma. ■

Outros anéis importantes que serão utilizados com frequência neste trabalho são: anel polinomial e anel quociente.

Definição 40 (anel polinomial)**.** *Sendo o polinômio P na forma $P(x) = a_0 + a_1x^1 + a_2x^2 + ... + a_nx^n$ com $n \in \mathbb{N}$, P é um anel polinomial se seus coeficientes $(a_0, a_1, a_2, ..., a_n)$ pertencem a um anel A. Denota-se então que $P \in A[x]$.*

Definição 41 (anel quociente)**.** *Seja um anel A e um ideal bilateral $I \subseteq A$. Tem-se o anel quociente A/I formado pelo conjunto de todas as classes de equivalência modulo I.*

Definição 42 (classes e relações de equivalência)**.** *Uma relação R sobre dois elementos $x, y \in A$ denotada por xRy, é uma relação de equivalência se satisfazer as seguintes propriedades:*

(i)reflexiva: xRx

(ii)simétrica: xRy então, yRx

(iii)transitiva: xRy e yRz então xRz

O conjunto de todos os elementos pertencentes a uma relação de equivalência R, é conhecido como classe de equivalência de R.

Definição 43 (ideais)**.** *Sejam A um anel e I um subconjunto não vazio de A. Dizemos que I é um ideal à esquerda de A se satisfazer as seguintes propriedades:*

(i)se $x, y, \in I$, então $x - y \in I$ e,

(ii)se $a \in A, x \in I$, então $ax \in I$.

Analogamente define-se um ideal à direita I de A como um subconjunto não vazio tal que:

(i)se $x, y, \in I$, então $x - y \in I$ e,

(ii)se $a \in A, x \in I$, então $xa \in I$.

Se I é um ideal simultaneamente à direita e à esquerda de R dizemos que I é um ideal (bilateral) de R.

Exemplo 16: Sendo o anel dos inteiros $\mathbb{Z}$ e o conjunto dos inteiros pares representado por $2\mathbb{Z} \subset \mathbb{Z}$ é um ideal de $\mathbb{Z}$, pois a subtração de qualquer inteiro par por outro inteiro par resulta em um inteiro par, e a multiplicação de um inteiro par por outro inteiro qualquer, resulta em um inteiro par. ■

Outro tipo ideal que deve ser mencionado, é o ideal polinomial na forma $\langle x^n + 1\rangle \subset \mathbb{Z}[x]$. Este ideal representa todos os polinômios múltiplos de $x^n + 1$ pertencentes a $\mathbb{Z}[x]$, visto que, sejam $P(x), Q(x) \in \langle x^n + 1\rangle$, $P(x)(x^n + 1) - Q(x)(x^n + 1) = (P(x) - Q(x))(x^n + 1)$ e seja $G(x) \notin \langle x^n + 1\rangle$, $P(x)(x^n + 1)G(x) = (P(x)G(x))(x^n + 1)$.

O anel quociente $\mathbb{Z}_p[x]/\langle x^n + 1\rangle$ é de grande importância para esse trabalho, esta estrutura é utilizada no esquema M-LWE empregado pelo Kyber. Seja $a, b \in \mathbb{Z}_p[x]$, um ideal $\langle x^n + 1\rangle \in \mathbb{Z}_p[x]$ e a relação de equivalência definida por $a \equiv b$ (**mod** $\langle x^n + 1\rangle$) se $a - b \in \langle x^n + 1\rangle$. O anel quociente $\mathbb{Z}_p[x]/\langle x^n + 1\rangle$ é uma classe de equivalência formada

pelo conjunto de todos os polinômios pertencentes a $\mathbb{Z}_p[x]$ módulo $x^n + 1$. Isto é, um polinômio pertencente a $\mathbb{Z}_p[x]/\langle x^n + 1\rangle$ é um polinômio que pertence ao anel $\mathbb{Z}_p[x]$ de grau menor que n com coeficientes entre 0 e $p - 1$.

Exemplo 17: seja o anel quociente $\mathbb{Z}_7[x]/\langle x^5 + 1\rangle$ e os polinômios $a(x) = 5 + 6x + x^3 - 2x^4 - 3x^5 \in \mathbb{Z}_7[x]/\langle x^5 + 1\rangle$ e $b(x) = 1 - x + 4x^2 + 3x^4 + x^5 \in \mathbb{Z}_7[x]/\langle x^5 + 1\rangle$. A operação de soma entre $a(x)$ e $b(x)$ é definida por:

$$\begin{aligned} c(x) &= a(x) + b(x) \\ c(x) &= (5 + 6x + x^3 - 2x^4 - 3x^5) + (1 - x + 4x^2 + 3x^4 + x^5) \\ c(x) &= 6 + 5x + 4x^2 + x^3 + x^4 - 2x^5 \end{aligned}$$

■

A operação de soma não altera o grau do polinômio resultante, entretanto a multiplicação polinomial altera. Desta forma, para que o resultado pertença ao anel quociente $\mathbb{Z}_7[x]/\langle x^5 + 1\rangle$, é necessário realizar uma divisão polinomial onde o resultado é o resto.

Exemplo 18: Sejam os mesmos polinômios do exemplo 17, a operação de multiplicação entre $a(x)$ e $b(x)$ é definida por:

$$\begin{aligned} c(x) &= a(x) * b(x) \\ c(x) &= (5 + 6x + x^3 - 2x^4 - 3x^5) * (1 - x + 4x^2 + 3x^4 + x^5) \\ c(x) &= 5 + x + 14x^2 + 25x^3 + 12x^4 + 26x^5 + x^6 - 9x^7 - 5x^8 - 11x^9 - 3x^{10} \\ c(x) &= c(x) \ \mathbf{mod}\ x^5 + 1 \\ c(x) &= -24 + 23x^2 + 30x^3 + 23x^4 \\ c(x) &= (-24 \ \mathbf{mod}\ 7) + (23 \ \mathbf{mod}\ 7)x^2 + (30 \ \mathbf{mod}\ 7)x^3 + (23 \ \mathbf{mod}\ 7)x^4 \\ c(x) &= 4 + 2x^2 + 2x^3 + 2x^4 \end{aligned}$$

■

Um polinômio que pertence ao anel quociente $\mathbb{Z}_7[x]/\langle x^5 + 1\rangle$ deve possui grau menor que 5 e coeficientes menores que 7. De modo prático, para limitar o grau de um polinômio que exceda grau 5, é realizada a operação **mod** $x^5 + 1$ sobre este polinômio, e para limitar os coeficientes é realizada a operação **mod** 7 em cada coeficiente do polinômio.

Definição 44 (corpos)**.** *Um corpo* $\mathbb{K}$ *é um anel comutativo com unidade, ou seja, possui elemento neutro para multiplicação, e todo elemento diferente do elemento neutro da adição (denotado por 0) possui um elemento inverso com relação à multiplicação, isso é* $\forall a \in \mathbb{K} - \{0\}, \exists b \in \mathbb{K}$ *tal que* $a.b = b.a = 1$.

O conjunto dos inteiros módulo p, sendo p um número primo, denotado por $\mathbb{Z}_p = \{0, 1, 2, ..., p-1\}$ é um corpo.

Exemplo 19: $\mathbb{Z}_3 = \{0, 1, 2\}$ é um corpo, pois $1 * 1$ **mod** $3 = 1$ e 2.2 **mod** $3 = 1$. ■

Contra-Exemplo 1: o conjunto $\mathbb{Z}_4 = \{0, 1, 2, 3\}$ não é um corpo, pois nenhum número pertencente a $\mathbb{Z}_4 - \{0\}$ multiplicado por 2 módulo 4 resulta em 1, ou seja, existe um elemento do conjunto $\mathbb{Z}_4$ que não possui inverso multiplicativo. ■

Definição 45. *Um módulo M sobre um anel unitário A (ou um A-módulo unitário) é um grupo abeliano (M, +) em conjunto com uma operação de um anel unitário A em M, que se escreve* $(a, v) \to av$*, satisfazendo as seguintes propriedades (FERNANDES; RICOU, 2004):*

(i) $a(v_1 + v_2) = av_1 + av_2, a \in A, v_1, v_2 \in M$;

(ii) $(a_1 + a_2)v = a_1v + a_2v, a_1, a_2 \in A, v \in M$;

(iii) $a_1(a_2v) = (a_1a_2)v, a_1, a_2 \in A, v \in M$;

(iv) $1v = v, v \in M$.

Um módulo se assemelha aos espaços vetoriais, com a diferença que os escalares dos espaços vetoriais pertencem a um corpo, enquanto os escalares de um módulo pertencem a um anel.

A.5 Espaços métricos

Quando for abordado o conceito de reticulados, estrutura na qual o algoritmo ML-KEM é baseado, serão necessários alguns conceitos envolvendo espaços métricos, visto que um reticulado é um espaço métrico. Alguns problemas envolvendo reticulados possuem sua dificuldade de resolução associada a uma métrica, desta forma, vê-se necessário uma introdução sobre espaços métricos.

Uma métrica em um conjunto M é uma função $d : M \times M \to \mathbb{R}^+$, essa função associa cada par de elementos $x, y \in M$ a um número real positivo $d(x, y)$, chamado distância de x a y. Seja x, y e $z \in M$, a função d precisa satisfazer as seguintes propriedades:

(i)$d(x, x) = 0$,

(ii)Se $x \neq y$ então $d(x, y) > 0$,

(iii)$d(x, y) = d(y, x)$ e

(iv)$d(x, z) \leq d(x, y) + d(y, z)$.

Alguns exemplos de métricas que podem ser utilizada para medir a distância entre dois elementos x, y de um conjunto qualquer são:

Métrica euclidiana:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Métrica de Manhattan:

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

Métrica de Chebyshev:

$$d(x, y) = \lim_{p \to \infty} \left(\sum_{i=1}^{n} (x_i - y_i)^p \right)^{1/p}$$

Métrica de Minkowski:

$$d(x, y) = \left(\sum_{i=1}^{n} (x_i - y_i)^p \right)^{1/p}$$

A métrica de Minkowski, também é conhecida como norma l_p, é uma generalização das três métricas anteriores: euclidiana $p = 2$, Manhattan $p = 1$ e com p tendendo ao infinito tem-se a métrica de Chebyshev.

Definição 46 (espaço métrico)**.** *Um espaço métrico é um par* (M, d)*, onde* M *é um conjunto e* d *é uma métrica em* M*.*

Exemplo 20: O espaço euclidiano $\mathbb{R}^n$ é um espaço métrico, onde os elementos de $\mathbb{R}^n$ são pontos $x = (x_1, x_2, ..., x_n)$. Com a distância definida pela métrica euclidiana. ■

Outras definições importantes que é necessário abordar são os conceitos de bolas e esferas. Seja a um ponto no espaço métrico M e $r > 0$ um número real, definimos:

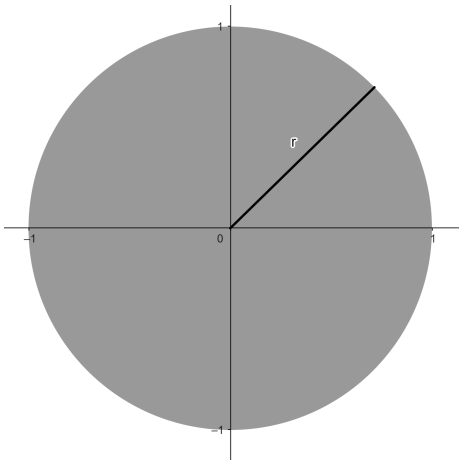
Definição 47 (bola aberta)**.** *A bola aberta de centro* a *e raio* r*, é o conjunto de pontos de* M *cuja distância ao ponto* a *é menor que* r*.* $B(a, r) = \{x \in M \mid d(x, a) < r\}$*.*

Definição 48 (bola fechada)**.** *A bola fechada de centro* a *e raio* r*, é o conjunto de pontos de* M *cuja distância ao ponto* a *é menor ou igual à* r*.* $B[a, r] = \{x \in M \mid d(x, a) \leq r\}$*.*

Definição 49 (esfera)**.** *A esfera de centro* a *e raio* r*, é o conjunto de pontos de* M *cuja distância ao ponto* a *é igual à* r*.* $S(a, r) = \{x \in M \mid d(x, a) = r\}$*.*

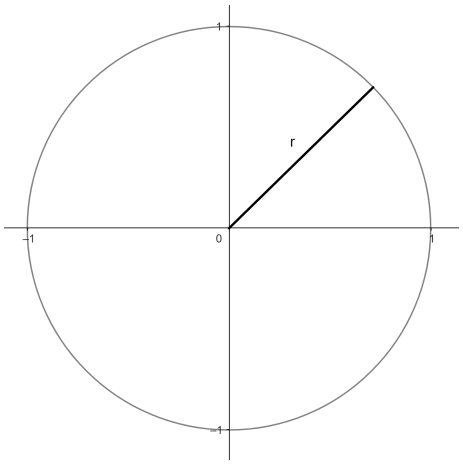
A Figura A.2 mostra um exemplo de uma bola no plano com raio 1, contendo todos os pontos em cinza. Já a Figura A.3 mostra um exemplo de uma esfera de raio 1. Lembrando que pode-se ter bolas e esferas de n dimensões, dependendo do espaço métrico que está sendo utilizado.

Figura A.2: Bola no plano.



Fonte: O autor.

Figura A.3: Esfera no plano.



Fonte: O autor.

Seja a um ponto pertencente ao espaço métrico M, dizemos que o ponto a é um ponto isolado se $M \cap B(a, r) = \{a\}$, para algum $r > 0$, isto é, se além do próprio ponto a, nenhum outro ponto de M está a uma distância inferior a r de a.

Definição 50 (espaço métrico discreto)**.** *Um espaço métrico M é dito discreto, se todo elemento pertencente a M for isolado.*

Exemplo 21: o espaço métrico do conjunto dos inteiros e distância euclidiana é um espaço métrico discreto, pois conseguimos definir um $r > 0$ tal que seja o raio de uma bola aberta com origem em quaisquer inteiro e que este inteiro, seja isolado. Já o conjunto $\{0, 1, 1/2, ..., 1/n\}$ não é discreto, pois o elemento 0 não é isolado para algum elemento do conjunto, ou seja, não conseguimos definir uma distância mínima entre os elementos desse conjunto. ■

B Demonstrações

Para facilitar a leitura do texto, todas as demonstrações de teoremas e isomorfismos estarão separados neste apêndice.

B.1 Teorema fundamental da aritmética

Teorema B.1.1 (teorema fundamental da aritmética)**.** *Todo número natural maior que 1 ou é primo, ou pode ser escrito como um produto de números primos.*

Demonstração.

Por indução em n.

1. *base da indução*: sendo $n = 2$ e como 2 é um número primo, a base está provada.
2. *hipótese de indução*: sendo um $k \geq 2$ qualquer, supomos que todo $i = 2, 3, ..., k$ é primo ou um produto de primos.
3. *passo indutivo*: provar a propriedade para $k + 1$. Caso $k + 1$ seja número primo, a propriedade é satisfeita. Caso $k + 1$ seja número composto então $k + 1 = n_1 n_2$ para $1 < n_1 < k + 1$ e $1 < n_2 < k + 1$. Aplicando a hipótese de indução em n_1 e n_2, temos que $k + 1$ é um produto de primos, satisfazendo a propriedade.

■

B.2 Isomorfismos aditivos sobre reticulados

Seja σ_1 e σ_1^{-1} os seguintes mapeamentos entre um reticulado de base de tamanho n e polinômios de grau $n-1$.

$$\begin{aligned}
\sigma_1 &: P[x] \to \mathbb{Z}^n \\
&: a_1 + a_2x + a_3x^2 + ... + a_nx^{n-1} \to (a_1, a_2, a_3, ..., a_n) \\
\sigma_1^{-1} &: \mathbb{Z}^n \to P[x] \\
&: (a_1, a_2, a_3, ..., a_n) \to a_1 + a_2x + a_3x^2 + ... + a_nx^{n-1}
\end{aligned}$$

Para provar que de fato σ_1 é um isomorfismo aditivo, deve-se demonstrar três propriedades:

(i)$\sigma_1(P[x] + Q[x]) = \sigma_1(P[x]) + \sigma_1(Q[x])$,

(ii)$\sigma_1(\sigma_1^{-1}) = \mathbb{Z}^n$ e

(ii)$\sigma_1^{-1}(\sigma_1) = P[x]$.

Demonstração do item (i):

$$\begin{aligned}
\sigma_1((a_1 + a_2x + a_3x^2 + ... + a_nx^{n-1}) + (b_1 + b_2x + b_3x^2 + ... + b_nx^{n-1})) &= \\
\sigma_1(a_1 + a_2x + a_3x^2 + ... + a_nx^{n-1}) + \sigma_1(b_1 + b_2x + b_3x^2 + ... + b_nx^{n-1})
\end{aligned}$$

$$\begin{aligned}
\sigma_1((a_1 + b_1) + (a_2 + b_2)x + (a_3 + b_3)x^2 + ... + (a_n + b_n)x^{n-1}) &= \\
\sigma_1(a_1 + a_2x + a_3x^2 + ... + a_nx^{n-1}) + \sigma_1(b_1 + b_2x + b_3x^2 + ... + b_nx^{n-1})
\end{aligned}$$

$$\begin{aligned}
(a_1 + b_1, a_2 + b_2, a_3 + b_3, ..., a_n + b_n) &= \\
(a_1, a_2, a_3, ..., a_n) + (b_1, b_2, b_3, ..., b_n)
\end{aligned}$$

$$\begin{aligned}
(a_1 + b_1, a_2 + b_2, a_3 + b_3, ..., a_n + b_n) &= \\
(a_1 + b_1, a_2 + b_2, a_3 + b_3, ..., a_n + b_n)
\end{aligned}$$

Demonstração do item (ii):

$$\begin{aligned}
\sigma_1(\sigma_1^{-1}(a_1, a_2, a_3, ..., a_n)) &= (a_1, a_2, a_3, ..., a_n) \\
\sigma_1(a_1 + a_2x + a_3x^2 + ... + a_nx^{n-1}) &= (a_1, a_2, a_3, ..., a_n) \\
(a_1, a_2, a_3, ..., a_n) &= (a_1, a_2, a_3, ..., a_n)
\end{aligned}$$

Demonstração do item (iii):

$$\begin{aligned}
\sigma_1^{-1}(\sigma_1(a_1 + a_2x + a_3x^2 + ... + a_nx^{n-1})) &= a_1 + a_2x + a_3x^2 + ... + a_nx^{n-1} \\
\sigma_1^{-1}((a_1, a_2, a_3, ..., a_n)) &= a_1 + a_2x + a_3x^2 + ... + a_nx^{n-1} \\
a_1 + a_2x + a_3x^2 + ... + a_nx^{n-1} &= a_1 + a_2x + a_3x^2 + ... + a_nx^{n-1}
\end{aligned}$$

Desta forma provamos que σ_1 é um isomorfismo aditivo, isto significa que $P[x]$ e $\mathbb{Z}^n$ se comportam da mesma forma com relação à operação de adição. Seja σ_2 e σ_2^{-1} os seguintes mapeamentos entre um reticulado de base de tamanho n e matrizes-linha com n colunas.

$$\begin{aligned}
\sigma_2 &: M_n \to \mathbb{Z}^n \\
&: [a_1\ a_2\ a_3\ ...\ a_n] \to (a_1, a_2, a_3, ..., a_n) \\
\sigma_2^{-1} &: \mathbb{Z}^n \to M_n \\
&: (a_1, a_2, a_3, ..., a_n) \to [a_1\ a_2\ a_3\ ...\ a_n]
\end{aligned}$$

Para provar que σ_2 é um isomorfismo aditivo, devemos garantir as mesmas três propriedades que provamos para σ_1, mas agora respeitando o domínio e contradomínio de σ_2.

Demonstração do item (i):

$$\begin{aligned}
\sigma_2([a_1\ a_2\ a_3\ ...\ a_n] + [b_1\ b_2\ b_3\ ...\ b_n]) &= \sigma_2([a_1\ a_2\ a_3\ ...\ a_n]) + \sigma_2([b_1\ b_2\ b_3\ ...\ b_n]) \\
\sigma_2([a_1 + b_1\ a_2 + b_2\ a_3 + b_3\ ...\ a_n + b_n]) &= \sigma_2([a_1\ a_2\ a_3\ ...\ a_n]) + \sigma_2([b_1\ b_2\ b_3\ ...\ b_n]) \\
(a_1 + b_1, a_2 + b_2, a_3 + b_3, ..., a_n + b_n) &= (a_1, a_2, a_3, ..., a_n) + (b_1, b_2, b_3, ..., b_n) \\
(a_1 + b_1, a_2 + b_2, a_3 + b_3, ..., a_n + b_n) &= (a_1 + b_1, a_2 + b_2, a_3 + b_3, ..., a_n + b_n)
\end{aligned}$$

Demonstração do item (ii):

$$\begin{aligned}
\sigma_2(\sigma_2^{-1}((a_1, a_2, a_3, ..., a_n))) &= (a_1, a_2, a_3, ..., a_n) \\
\sigma_2([a_1\ a_2\ a_3\ ...\ a_n]) &= (a_1, a_2, a_3, ..., a_n) \\
(a_1, a_2, a_3, ..., a_n) &= (a_1, a_2, a_3, ..., a_n)
\end{aligned}$$

Demonstração do item (iii):

$$\begin{aligned}\sigma_2^{-1}(\sigma_2([a_1\ a_2\ a_3\ ...\ a_n])) &= [a_1\ a_2\ a_3\ ...\ a_n]\\ \sigma_2^{-1}((a_1, a_2, a_3, ..., a_n)) &= [a_1\ a_2\ a_3\ ...\ a_n]\\ [a_1\ a_2\ a_3\ ...\ a_n] &= [a_1\ a_2\ a_3\ ...\ a_n]\end{aligned}$$