UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
INSTITUTO TÉRCIO PACITTI DE APLICAÇÕES E PESQUISAS COMPUTACIONAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

CHARLES FIGUEREDO DE BARROS

AUTENTICAÇÃO E CRIPTOGRAFIA PÓS-QUÂNTICA BASEADA EM RETICULADOS

Rio de Janeiro
2014

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
INSTITUTO TÉRCIO PACITTI DE APLICAÇÕES E PESQUISAS COMPUTACIONAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

CHARLES FIGUEREDO DE BARROS

AUTENTICAÇÃO E CRIPTOGRAFIA PÓS-QUÂNTICA BASEADA EM RETICULADOS

Dissertação de Mestrado submetida ao Corpo Docente do Departamento de Ciência da Computação do Instituto de Matemática, e Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários para obtenção do título de Mestre em Informática.

Orientador: Luis Menasché Schechter

Rio de Janeiro
2014

B277 Barros, Charles Figueredo de

Autenticação e criptografia pós-quântica baseada em reticulados / Charles Figueredo de Barros. – 2014.

124 f.: il.

Dissertação (Mestrado em Informática) – Universidade Federal do Rio de Janeiro, Instituto de Matemática, Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Programa de Pós-Graduação em Informática, Rio de Janeiro, 2014.

Orientador: Luis Menasché Schechter.

1. Criptografia de Chave Pública. 2. Reticulados. 3. Autenticação – Teses. I. Schechter, Luis Menasché (Orient.). II. Universidade Federal do Rio de Janeiro, Instituto de Matemática, Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Programa de Pós-Graduação em Informática. III. Título

CDD:

CHARLES FIGUEREDO DE BARROS

AUTENTICAÇÃO E CRIPTOGRAFIA PÓS-QUÂNTICA BASEADA EM RETICULADOS

Dissertação de Mestrado submetida ao Corpo Docente do Departamento de Ciência da Computação do Instituto de Matemática, e Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários para obtenção do título de Mestre em Informática.

Aprovado em: Rio de Janeiro, _____ de _____ de _____.

_____

Luis Menasché Schechter (Orientador), D.Sc., UFRJ

_____

Severino Collier Coutinho, Ph.D., UFRJ

_____

Luziane Ferreira de Mendonça, D.Sc., UFRJ

_____

Juliana Vianna Valerio, D.Sc., UFRJ

Para Sidnei e Vanderléia.

AGRADECIMENTOS

Jean de la Bruyère afirmou que “não há no mundo exagero mais belo que a gratidão”. Assim ficarei feliz se, porventura, eu for *exagerado* em meus agradecimentos. Em primeiro lugar agradeço a meus pais, porque a eles devo tudo que sou. Também sou grato ao Professor João Luiz Lincoln, por ter me oferecido uma oportunidade que poucos seriam capazes de oferecer. Ao Corpo Docente do Instituto de Matemática da UFRJ, por ter me concedido o privilégio de uma educação superior de qualidade, em um país onde a educação ainda é um problema profundamente enraizado. Ao Professor Severino Collier Coutinho por ser, além de um profissional altamente capacitado, um grande amigo e uma pessoa muito querida. Sua dedicação, profissionalismo e amizade foram marcantes durante o tempo em que ele foi meu orientador de Iniciação Científica, e sinto-me honrado em poder desfrutar dessas virtudes do Professor Collier até os dias de hoje. Expresso minha gratidão também a todo o Corpo Docente do Departamento de Ciência da Computação e do Programa de Pós-Graduação em Informática. É difícil encontrar um ambiente acadêmico que ofereça uma formação de tão alto nível e seja, ao mesmo tempo, tão prazeroso. Também não posso deixar de expressar meus agradecimentos aos colegas do LC3, que me presentearam com sua amizade e senso de humor inigualáveis. Agradeço também ao meu orientador, Professor Luis Menasché Schechter, por sua paciência, dedicação, companheirismo e pela confiança depositada em mim. Espero poder retribuir à altura com este trabalho e com os trabalhos vindouros. Finalmente agradeço à CAPES pelo suporte financeiro.

RESUMO

Barros, Charles Figueredo de. **Autenticação e criptografia pós-quântica baseada em reticulados**. 2014. 124 f. Dissertação (Mestrado em Informática) - PPGI, Instituto de Matemática, Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2014.

A segurança dos sistemas atuais de criptografia de chave pública está diante de uma ameaça latente: um computador quântico capaz de executar os algoritmos de fatoração e cálculo do logaritmo discreto. Se ele for criado, os sistemas vigentes, incluindo o RSA, terão de ser substituídos por métodos de criptografia denominados *pós-quânticos*, já que nem mesmo os computadores quânticos seriam capazes de quebrá-los eficientemente usando algoritmos conhecidos.

Neste trabalho fazemos um estudo de um desses métodos pós-quânticos: o GGH. Criado em 1997 por Oded Goldreich, Shafi Goldwasser (uma das ganhadoras do último Prêmio Turing) e Shai Halevi, este sistema tem sua segurança baseada na intratabilidade de certos problemas da teoria de reticulados. Também será feito um estudo do GGH-YK, uma variante do GGH proposta recentemente.

As contribuições deste trabalho podem ser divididas em três partes: na primeira, será mostrado que, na prática, o GGH-YK não é diferente do GGH; em seguida, será proposta uma nova variante, batizada de GGHYK-M, resultante de algumas modificações no GGH-YK. Finalmente algumas possibilidades para um protocolo de autenticação baseado no GGHYK-M serão discutidas.

Palavras-chave: Criptografia de Chave Pública, Reticulados, Autenticação.

ABSTRACT

Barros, Charles Figueredo de. **Autenticação e criptografia pós-quântica baseada em reticulados**. 2014. 124 f. Dissertação (Mestrado em Informática) - PPGI, Instituto de Matemática, Instituto Tércio Pacitti, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2014.

The security of the current public-key cryptosystems is facing a latent threat: a quantum computer capable of running the algorithms for integer factorization and discrete logarithm computation. If such a computer is built, existing cryptosystems, including RSA, will have to be replaced by the so-called *post-quantum* encryption methods, since not even quantum computers would be able to break them efficiently using known algorithms.

In this work we present a study of one of these post-quantum methods, called GGH. Created in 1997 by Oded Goldreich, Shafi Goldwasser (one of the winners of the last A. M. Turing Award) and Shai Halevi, the security of this cryptosystem is based on the intractability of certain problems from the theory of lattices. It will also be presented a study of GGH-YK, a recently proposed variant of GGH.

The contributions of this work can be divided into three parts: in the first part, it will be shown that, in practice, GGH-YK is not different from the original GGH; then a new variant, called GGHYK-M, will be proposed as a result of some modifications on GGH-YK. Finally, some possibilities for an authentication protocol based on GGHYK-M will be discussed.

Keywords: Public-Key Cryptography, Lattices, Authentication.

LISTA DE FIGURAS

Figura 1.1: Máquina Enigma. [Fonte: http://www.bbc.co.uk/history/topics/enigma]. . . 18
Figura 1.2: Esquema de criptografia simétrica. A chave secreta é capaz tanto de encriptar quanto de decriptar. 20
Figura 1.3: Esquema de criptografia assimétrica. Duas chaves são necessárias. 22
Figura 1.4: O RSA é utilizado como algoritmo de encriptação em páginas seguras da web, como a página de login do GMail. 23
Figura 1.5: DWave, o primeiro computador quântico a ser comercializado. [Fonte: D-Wave Systems]. 29

Figura 2.1: Ortogonalização de Gram-Schmidt em duas dimensões. 39

Figura 3.1: Domínio fundamental de um reticulado em duas dimensões. 45
Figura 3.2: Particionamentos induzidos por quatro bases de um mesmo reticulado. A área do domínio fundamental de todos os particionamentos é constante, graças à invariância do determinante do reticulado. 46

LISTA DE TABELAS

Tabela 1.1: Chave de encriptação/decriptação da cifra de César. Para encriptar uma mensagem, basta substituir cada letra da linha superior pela letra correspondente na linha inferior, e para decriptar, basta fazer o processo inverso. . . . 16
Tabela 1.2: Tabela de combinações da cifra de Vigenère. 17
Tabela 1.3: Encriptação da mensagem "Encontre-me no parque às seis" com a cifra de Vigenère original, utilizando a chave inicial X. 17
Tabela 1.4: Encriptação da mensagem "Encontre-me no parque às seis" com a cifra de Vigenère modificada, utilizando a palavra CHAVE como chave. 18
Tabela 1.5: Encriptação de uma mensagem com a cifra de Vernam. 18

Tabela 4.1: Estimativa dos tamanhos das chaves públicas (em kB) no GGH, sem a FHN e com a FHN. 80

Tabela 7.1: Desempenho do LLL contra as chaves públicas do GGH-YK. 110
Tabela 7.2: Desempenho do BKZ contra as chaves públicas do GGH-YK. 110
Tabela 7.3: Desempenho do LLL contra as chaves públicas do GGHYK-M. 110
Tabela 7.4: Desempenho do BKZ contra as chaves públicas do GGHYK-M. 111
Tabela 7.5: Comparação entre os tamanhos das chaves públicas (em kB) no GGH, sem a FHN e com a FHN. 112
Tabela 7.6: Tamanho das chaves públicas do GGHYK-M. 112
Tabela 7.7: Tempo em segundos da geração de chaves no GGHYK-M. 113
Tabela 7.8: Tempo em segundos de encriptação e decriptação no GGHYK-M. 114
Tabela 7.9: Tempo em segundos da decriptação no GGHYK-M, com a inversa da chave secreta previamente armazenada. 115
Tabela 7.10: Comparação entre o tamanho em bits das mensagens e dos respectivos criptogramas no GGHYK-M. 115

SUMÁRIO

ANTES DE COMEÇARMOS... 13

1 CRIPTOGRAFIA: ARTE E CIÊNCIA 15
1.1 Mlecchita Vikalpa: a 44ª arte 15
1.2 Lucifer e Rijndael 19
1.3 Um cofre em praça pública e como compartilhar segredos . . . 20
1.4 Assine aqui, por favor 26
1.5 A revolução dos *quanta* 27
1.6 O que já foi feito... 30
1.7 ...e o que há por fazer 31

2 BREVE REVISÃO DE ÁLGEBRA LINEAR 33
2.1 Espaços vetoriais 33
2.1.1 Normas e produtos internos 34
2.1.2 Bases 37
2.1.3 Bases Ortogonais e Ortonormais 38
2.2 Autovetores e Autovalores 40

3 RETICULADOS 42
3.1 Bases de um Reticulado 42
3.1.1 Domínios Fundamentais 44
3.2 Problemas Computacionais 49
3.3 Bases Reduzidas 51
3.3.1 Redução em Duas Dimensões 51
3.3.2 Bases de Gram-Schmidt e o Volume de um Reticulado 52
3.3.3 Bases Lovász-Reduzidas 53
3.3.4 O Algoritmo LLL 57
3.3.5 O LLL Inteiro 64
3.3.6 Bases Korkine-Zolotareff-Reduzidas e o algoritmo BKZ 69

4 OS SISTEMAS GGH E GGH-YK 73
4.1 Empregando o CVP como problema subjacente 73
4.2 Geração de chaves, encriptação e decriptação 74
4.3 Redução de bases e a segurança do GGH 76
4.4 Novos rumos para o GGH 80
4.5 Detalhes do GGH-YK 84

5 GGHYK-M: UMA VARIANTE DO GGH-YK 89
5.1 Melhorias no esquema de Yoshino e Kunihiro 89
5.2 Construindo uma nova variante 90
5.3 M-matrizes 96

6 PROTOCOLOS DE AUTENTICAÇÃO BASEADOS NO GGHYK-M . 98
6.1 Visão geral dos protocolos de autenticação 98
6.2 Autenticação por senhas 100
6.3 Autenticação por *challenge-response* 102
6.4 Empregando o GGHYK-M em um protocolo de autenticação . 103
6.4.1 Autenticação mútua com o GGHYK-M 105
6.5 Considerações finais sobre autenticação 107

7 RESULTADOS, CONCLUSÕES E TRABALHOS FUTUROS 108
7.1 Eficiência dos algoritmos de redução 108
7.2 Tamanho das bases na Forma Hermitiana Normal 112
7.3 Desempenho do GGHYK-M 113
7.4 GGH-YK e GGHYK-M 115
7.5 Trabalhos futuros 117

REFERÊNCIAS 118

ANTES DE COMEÇARMOS...

> "Nós nunca nos realizamos. Somos dois abismos - um poço fitando o céu." (Fernando Pessoa)

A criptografia, como qualquer campo do conhecimento, tem o seu jargão próprio. O objetivo deste capítulo é explicar o significado de alguns termos que serão utilizados com frequência ao longo deste trabalho. Outras expressões, que exigem uma descrição mais formal e detalhada, serão apresentadas no decorrer do texto.

- Mensagem: refere-se a qualquer informação que deva ser transmitida ou armazenada.
- Encriptação: é o processo pelo qual uma mensagem é transformada, de modo que se torne ilegível para pessoas não autorizadas a ler o seu conteúdo.
- Criptograma: é o resultado da encriptação de uma mensagem.
- Decriptação: refere-se ao processo de recuperar uma mensagem a partir de um criptograma.
- Chave: termo genérico que designa algum tipo de informação utilizada para encriptação/decriptação de mensagens. A natureza específica da chave pode ser um número, uma matriz, uma string de bits, etc.
- Criptossistema: termo genérico que se refere a um conjunto que deve conter, necessariamente, um algoritmo de encriptação, um algoritmo de decriptação e uma ou mais chaves.

- Decifrar: refere-se ao processo de recuperar uma mensagem a partir de um criptograma, porém sem o conhecimento prévio da chave. Geralmente isto é feito por um usuário não autorizado (um *cracker*, por exemplo).

- Quebrar: diz-se que um criptossistema foi quebrado se nele foi descoberta alguma falha intrínseca, permitindo que usuários não autorizados decifrem criptogramas ou descubram as chaves do sistema.

- Ataque: um ataque é uma tentativa de quebrar um criptossistema, o que pode ou não envolver a interceptação de criptogramas.

- Alice, Bernardo e Eva: personagens que ilustram os exemplos de troca de mensagens secretas. Alice e Bernardo são os usuários legítimos, e Eva faz o papel de adversário que ataca o criptossistema utilizado por eles.

- Codificação: processo pelo qual uma mensagem é escrita em algum alfabeto específico, com a propósito de facilitar o seu armazenamento ou envio, ou a fim de prepará-la para a encriptação. Diferente desta, a codificação não tem nenhum propósito de esconder o conteúdo da mensagem.

- Decodificação: processo inverso da codificação.

- Canal de comunicação: é o meio físico através do qual uma mensagem é transmitida. Ex: linhas telefônicas, cabos de rede, etc.

1 CRIPTOGRAFIA: ARTE E CIÊNCIA

> "De sonhar ninguém se cansa, porque sonhar é esquecer, e esquecer não pesa e é um sono sem sonhos em que estamos despertos." (Fernando Pessoa)

Neste capítulo será feita uma breve revisão da história da criptografia, desde os primórdios dos sistemas de escrita secreta até os modernos criptossistemas utilizados na Internet. O objetivo é situar a criptografia pós-quântica neste cenário.

1.1 Mlecchita Vikalpa: a 44ª arte

A criptografia é a arte (e ciência) de escrever mensagens secretas. Em [1] relata-se que as primeiras formas de escrita secreta remontam ao antigo Egito, e que até mesmo o Kama-Sutra lista a escrita secreta (*Mlecchita Vikalpa*) como uma das 64 artes[1] que as mulheres deveriam aprender.

Saindo do antigo Egito e indo até Roma, encontramos um dos mais conhecidos exemplos históricos de escrita secreta: a *cifra de César*. Nela, a encriptação consiste em substituir cada letra de uma mensagem pela letra situada três posições à frente (de forma cíclica). Desta forma, "A" é substituído por "D", "B" por "E", e assim por diante (veja a Tabela 1.1). Para a decriptação, portanto, basta aplicar o processo inverso, substituindo cada letra por aquela situada três posições atrás no alfabeto, também de forma cíclica.

[1] A *Mlecchita Vikalpa* é a 44ª arte listada no Kama-Sutra.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Tabela 1.1: Chave de encriptação/decriptação da cifra de César. Para encriptar uma mensagem, basta substituir cada letra da linha superior pela letra correspondente na linha inferior, e para decriptar, basta fazer o processo inverso.

Criada no século XVI, a *cifra de Vigenère* é outro exemplo histórico de sistema de escrita secreta. Para utilizá-la, deve-se recorrer à Tabela 1.2. As letras da mensagem devem ser procuradas na linha superior, e as da chave na coluna mais à esquerda. A letra correspondente do criptograma será aquela contida na interseção entre a linha da letra da chave e a coluna da letra da mensagem.

Por exemplo, se a letra da mensagem for `F` e a letra da chave for `C`, a letra correspondente do criptograma será `H` (interseção da linha `C` com a coluna `F`). Para decriptar, basta procurar a letra `H` na linha `C` e verificar em qual coluna ela se encontra.

Originalmente, a cifra de Vigenère foi concebida como um sistema de *auto-chave*, isto é, que utiliza a própria mensagem como chave, além de uma letra secreta que funciona como chave inicial. Na encriptação, combina-se a primeira letra da mensagem com a chave inicial, a segunda letra com a primeira, e assim por diante (veja a Tabela 1.3).

Cerca de três séculos depois de ter sido concebida, a cifra de Vigenère foi reinventada, assumindo a forma pela qual é conhecida nos dias de hoje. A tabela de combinações ainda é a mesma, porém a mensagem não é mais utilizada como chave. No lugar dela, utiliza-se uma palavra secreta, que é repetida sob a mensagem até atingir o comprimento desta (veja a Tabela 1.4).

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Tabela 1.2: Tabela de combinações da cifra de Vigenère.

Mensagem	E	N	C	O	N	T	R	E	M	E	N	O	P	A	R	Q	U	E	A	S	S	E	I	S
Chave	X	E	N	C	O	N	T	R	E	M	E	N	O	P	A	R	Q	U	E	A	S	S	E	I
Criptograma	B	R	P	Q	B	G	K	V	Q	Q	R	B	D	P	R	H	K	Y	E	S	K	W	M	A

Tabela 1.3: Encriptação da mensagem "Encontre-me no parque às seis" com a cifra de Vigenère original, utilizando a chave inicial X.

Em 1917, Gilbert Vernam, da AT&T[2], desenvolveu o que hoje conhecemos como *cifra de Vernam*. Nela, mensagem e chave devem ser codificadas em strings

[2]American Telephone and Telegraph Company.

Mensagem	E N C O N T R E M E N O P A R Q U E A S S E I S
Chave	C H A V E C H A V E C H A V E C H A V E C H A V
Criptograma	G U C J R V Y E H I P V P V V S B E V W U L I N

Tabela 1.4: Encriptação da mensagem "Encontre-me no parque às seis" com a cifra de Vigenère modificada, utilizando a palavra CHAVE como chave.

binárias do mesmo comprimento. A encriptação é realizada bit a bit, empregando-se a operação binária XOR (*ou exclusivo*) entre os bits da mensagem e da chave. Para decriptar, basta aplicar a mesma operação entre os bits do criptograma e da chave (veja a Tabela 1.5). Se cada chave for escolhida de forma aleatória e utilizada uma única vez, a cifra de Vernam é também chamada de *One Time Pad* (OTP).

Mensagem	1 0 0 1 1 1 0 1 1 0 1 1 0 1 1 0
Chave	1 1 0 0 0 1 1 0 1 1 0 0 0 0 1 0
Criptograma	0 1 0 1 1 0 1 1 0 1 1 1 0 1 0 0

Tabela 1.5: Encriptação de uma mensagem com a cifra de Vernam.

Também ficaram conhecidas as *máquinas de rotor*, utilizadas até o fim da Segunda Guerra Mundial. Tais máquinas possuíam sistemas de rotores capazes de realizar substituições sucessivas das letras da mensagem, produzindo assim os criptogramas. Sem dúvida, a máquina de rotor mais famosa da História foi a *Enigma* (Figura 1.1), utilizada pelos nazistas durante a Segunda Guerra Mundial. Uma das etapas decisivas para o desfecho do conflito foi a criptoanálise das transmissões secretas nazistas, encriptadas com a Enigma e interceptadas pelos Aliados[3].

Figura 1.1: Máquina Enigma. [Fonte: http://www.bbc.co.uk/history/topics/enigma].

[3]Este trabalho de criptoanálise teve a contribuição de Alan Turing.

1.2 Lucifer e Rijndael

Com o uso crescente dos meios eletrônicos de comunicação, surgiu a necessidade de criar um padrão para o *design* de criptossistemas. Em 1973, o *National Bureau of Standards* (NBS), órgão do governo norte-americano, fez uma requisição oficial para que fosse estabelecido um padrão federal de encriptação. Nenhum criptossistema foi submetido. Após uma nova requisição oficial, em 1974, a IBM submeteu uma variante da cifra *Lucifer*, e esta foi adotada em 1976 como o primeiro padrão para encriptação de dados (*Data Encryption Standard* ou simplesmente DES). O DES passou a ser obrigatório em aplicações governamentais para proteção de dados em computadores.

Lucifer é uma *cifra de bloco*, pois opera sobre blocos de comprimento fixo de 128 bits, empregando chaves do mesmo tamanho (ao estabelecer o DES, a Agência de Segurança Nacional reduziu o tamanho do bloco para 64 bits e o comprimento da chave para 56 bits). O DES opera através de um ciclo de 16 repetições de uma rede de Feistel[4], um intricado circuito que transpõe e substitui os bits da mensagem utilizando uma também intricada função pseudo-aleatória (calculada por um algoritmo determinístico, mas cuja distribuição é indistinguível de uma distribuição verdadeiramente aleatória). Além da rede de Feistel e da função pseudo-aleatória, Lucifer emprega um mecanismo de *expansão*, que transforma a chave do sistema em 16 chaves distintas, utilizadas nas iterações da rede de Feistel.

Anos depois, em 1997, o DES foi finalmente quebrado por busca exaustiva. O antigo NBS, agora conhecido como NIST (*National Institute of Standards and Technology*) fez uma nova requisição para submissões, e em 2001 anunciou a *cifra de Rijndael* como novo padrão de encriptação de dados, desta vez denominado AES (*Advanced Encryption Standard*).

[4]Em referência a Horst Feistel, desenvolvedor da versão original de Lucifer.

Desenvolvida na Bélgica, a cifra de Rijndael também é uma cifra de bloco de 128 bits, podendo empregar chaves de até 256 bits. O seu funcionamento também envolve um mecanismo de expansão de chave, além de uma rede de substituições e permutações, em um ciclo de 10 repetições.

Há muitos detalhes envolvidos no funcionamento do DES e do AES, e uma descrição plena foge do escopo deste trabalho. Para uma abordagem completa, pode-se consultar [2] e [3].

1.3 Um cofre em praça pública e como compartilhar segredos

Até meados da década de 70, os sistemas de criptografia eram pensados sempre de maneira *simétrica*, pois a mesma chave usada para encriptar mensagens era utilizada também para decriptá-las (Figura 1.2). Assim, tais chaves deveriam ser sempre secretas, sendo conhecidas apenas pelas partes que desejavam trocar mensagens criptografadas. O DES e o AES são exemplos de sistemas de criptografia simétrica (cifras).

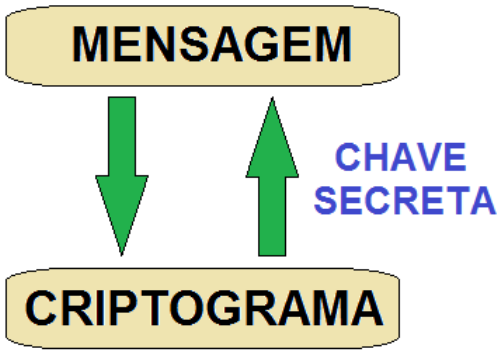

Figura 1.2: Esquema de criptografia simétrica. A chave secreta é capaz tanto de encriptar quanto de decriptar.

De maneira um pouco mais formal, uma cifra é constituída por dois algoritmos, E e D, utilizados para encriptação e decriptação respectivamente. O algoritmo E toma como entrada uma chave k e uma mensagem m, retornando um criptograma c. Para decriptar c, deve-se utilizar a mesma chave k como parâmetro para o algoritmo D. Traduzindo em equações, temos

$$\begin{cases} c := E(k, m), \\ D(k, c) = m. \end{cases} \tag{1.1}$$

Este paradigma traz à tona um problema se Alice quiser receber mensagens encriptadas, não apenas de Bernardo, mas de uma quantidade enorme de usuários, de modo que nenhum deles seja capaz de ver o conteúdo das mensagens enviadas pelos outros. Neste caso, Alice deveria estabelecer uma chave secreta diferente para cada um desses usuários, o que pode se tornar extremamente custoso.

Isto nos faz imaginar se é possível que ela compartilhe uma única chave com todos os usuários, e ainda assim garanta que somente ela seja capaz de decriptar qualquer mensagem. Diffie e Hellman responderam positivamente a esta questão ao desenvolverem o paradigma da *criptografia assimétrica* [4], ou *criptografia de chave pública*, que emprega duas chaves: uma chave secreta, utilizada na decriptação, e uma chave pública, empregada na encriptação de mensagens. Com o conhecimento apenas da chave pública, não é possível realizar a decriptação, enquanto que a chave secreta não é capaz de realizar a encriptação (Figura 1.3). Além disso, deve ser computacionalmente inviável descobrir a chave secreta a partir da chave pública.

Um sistema de chave pública, tomando de empréstimo a analogia apresentada em [5], é como um cofre aberto colocado em praça pública. A senha do cofre (a chave secreta) é conhecida apenas por Alice, mas qualquer usuário pode colocar uma mensagem dentro do cofre e trancá-lo (encriptar a mensagem). É muito difícil abrir o cofre sem o conhecimento da senha, ou seja, é difícil decriptar uma mensagem sem o conhecimento da chave secreta. Além disso, é muito difícil descobrir a senha,

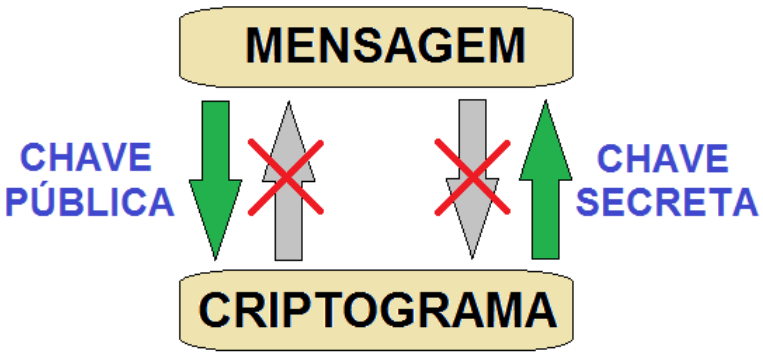

Figura 1.3: Esquema de criptografia assimétrica. Duas chaves são necessárias.

isto é, a chave secreta.

O segredo da criptografia assimétrica está nas chamadas *funções trapdoor seguras* (FTS). Informalmente, uma FTS é uma função que pode ser eficientemente calculada, mas não pode ser invertida a menos que se conheça alguma informação especial. No contexto de um criptossistema de chave pública, esta informação especial é a chave secreta.

Numa descrição formal, uma FTS é descrita por um algoritmo G que gera aleatoriamente pares (k_s, k_p) (chave secreta e chave pública, respectivamente), e uma função F tal que:

1. $F(k_p, x)$ é de *mão única* para todo x, isto é, não pode ser invertida eficientemente sem o conhecimento de k_s;

2. $F^{-1}(k_s, F(k_p, x)) = x$, para todo par (k_s, k_p) e para todo x.

A fim de garantir a segurança de uma função *trapdoor*, deve-se mostrar que invertê-la sem o conhecimento da chave secreta, ou descobrir a chave secreta, é tão difícil quanto resolver algum problema computacionalmente intratável. Este será dito en-

tão o *problema subjacente* à FTS.

Vale ressaltar que Diffie e Hellman desenvolveram o conceito de criptografia assimétrica de forma teórica, já que não apresentaram um criptossistema propriamente dito. De qualquer forma, eles plantaram uma valiosa semente, que rendeu frutos pouco depois com o RSA [6], que se tornou um dos mais populares criptossistemas de chave pública, sendo amplamente utilizado na *Internet* até os dias de hoje (Figura 1.4).

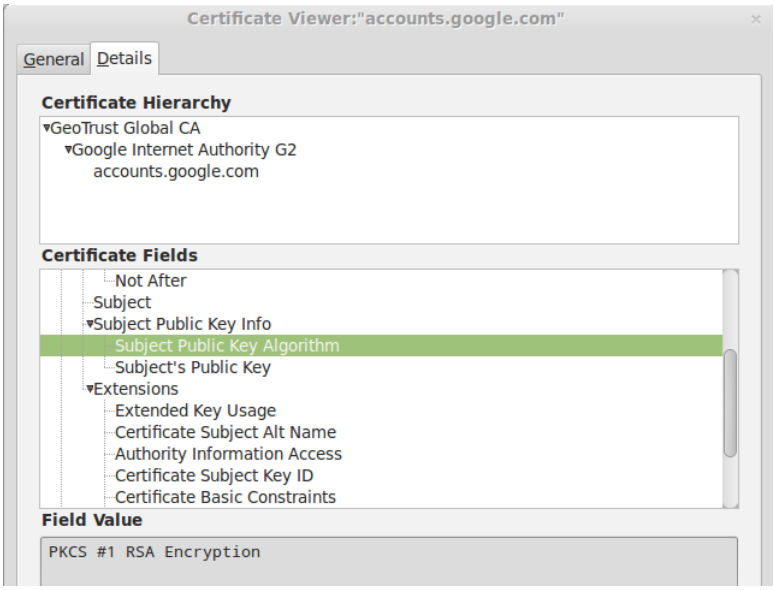

Figura 1.4: O RSA é utilizado como algoritmo de encriptação em páginas seguras da web, como a página de login do GMail.

Para gerar um par de chaves RSA, escolhem-se dois primos distintos, digamos p e q, e obtém-se o *módulo RSA*, dado por $N = pq$. Calculam-se então os inteiros e e d tais que

$$ed \equiv 1 \pmod{\varphi(N)}, \tag{1.2}$$

em que $\varphi(N) = (p-1)(q-1)$ é a função totiente de Euler. As chaves secreta e

pública são definidas então da seguinte maneira:

$$\begin{cases} k_s = (N, d), \\ k_p = (N, e). \end{cases} \tag{1.3}$$

Uma vez estabelecidas as chaves, podemos definir a FTS RSA, dada por

$$\begin{cases} \text{RSA}(k_p, x) = x^e \pmod{N}, \\ \text{RSA}^{-1}(k_s, y) = y^d \pmod{N}. \end{cases} \tag{1.4}$$

Pela congruência (1.2), é possível concluir que, de fato, o RSA satisfaz a propriedade 2 de uma FTS, ou seja,

$$\text{RSA}^{-1}(k_s, \text{RSA}(k_p, x)) = x.$$

Quanto à segurança, sabe-se que descobrir o expoente secreto d é tão difícil quanto fatorar o módulo N. Não se sabe ainda se existe uma maneira de inverter a função RSA sem o conhecimento de d. Enquanto isso, a única maneira conhecida de quebrar o sistema é através da *fatoração de inteiros*, que é um problema computacionalmente intratável. Ainda assim, existem muitos detalhes que devem ser levados em conta a fim de garantir a segurança efetiva do RSA. Em [7] é feita uma abordagem mais detalhada do assunto.

A criptografia assimétrica não foi a única semente plantada por Diffie e Hellman. Eles também propuseram um esquema de compartilhamento de chaves, baseado na intratabilidade de outro problema matemático: o *cálculo do logaritmo discreto*. Seja G um grupo. O *problema do logaritmo discreto* consiste em, dados $g, h \in G$, encontrar x tal que $g^x = h$.

Suponha que Alice e Bernardo desejam compartilhar uma chave secreta através de um canal de comunicação inseguro. Alice escolhe então um número primo p e um inteiro $g \in \{1, \cdots, p-1\}$. Estes dois valores são enviados para Bernardo. Em seguida, Alice escolhe um inteiro secreto $a \in \{1, \cdots, p-1\}$ e envia $A = g^a \pmod{p}$ para Bernardo. De maneira similar, Bernardo escolhe um inteiro secreto

$b \in \{1, \cdots, p-1\}$ e envia $B = g^b \pmod{p}$ para Alice. Finalmente, Alice calcula $B^a = g^{ab} \pmod{p}$, e Bernardo calcula $A^b = g^{ab} \pmod{p}$. O valor $g^{ab} \pmod{p}$ fica valendo como chave secreta entre eles.

Se Eva interceptar as mensagens trocadas entre Alice e Bernardo, tudo que ela vê são os valores p, g, $g^a \pmod{p}$ e $g^b \pmod{p}$. Se ela for capaz de resolver o problema do logaritmo discreto para g, g^a e g^b em $\mathbb{Z}_p$, ela determina os valores de a e b e descobre a chave secreta. Para a sorte de Alice e Bernardo, isto é muito difícil de fazer se p for suficientemente grande e g tiver ordem alta.

Esta é justamente a desvantagem do esquema: para alcançar níveis aceitáveis de segurança, o valor de p deve ser muito grande (da ordem de 600 dígitos), o que compromete a eficiência da troca de chaves. Este problema pode ser facilmente contornado se reinterpretarmos o esquema sobre um outro *cenário*: uma *curva elíptica*.

Uma curva elíptica E sobre um corpo finito $\mathbb{F}_p$ (denotada por $E(\mathbb{F}_p)$) é o conjunto das soluções (x, y) em $\mathbb{F}_p$ da equação

$$y^2 = x^3 + Ax + B,$$

em que $A, B \in \mathbb{F}_p$ satisfazem $4A^3 + 27B^2 \neq 0$, além de um ponto extra, denominado *ponto no infinito* e representado por $\mathcal{O}$. É possível mostrar que $E(\mathbb{F}_p)$ possui estrutura de grupo, com uma operação de *soma de pontos* especialmente definida. O *problema do logaritmo discreto sobre curvas elípticas* consiste em, dados os pontos $P, Q \in E(\mathbb{F}_p)$, determinar n tal que $Q = nP$. Neste contexto, nP representa o ponto P *somado* com ele mesmo n vezes.

Na versão do esquema de troca de chaves sobre curvas elípticas, Alice escolhe uma curva elíptica E e um ponto $P \in E$. Estas informações são compartilhadas com Bernardo. Ela escolhe então um inteiro secreto a, enviando aP para ele, e ele

escolhe um inteiro secreto b, enviando bP para ela. Para estabelecer a chave secreta, Alice calcula $a(bP) = abP$, e Bernardo calcula $b(aP) = abP$.

Se Eva for capaz de, dados os pontos P, aP e bP, descobrir os valores de a e b (ou seja, de calcular o logaritmo discreto sobre a curva E), ela pode então descobrir a chave secreta de Alice e Bernardo. Felizmente para eles, o problema do logaritmo discreto sobre curvas elípticas também é computacionalmente intratável. No entanto, fazer as contas sobre curvas elípticas tem a vantagem de ser mais eficiente, por exigir o uso de chaves muito menores.

Vale ressaltar que o emprego de curvas elípticas na criptografia de chave pública foi proposto, de forma independente, em [8] e [9].

1.4 Assine aqui, por favor

Além de permitir a transmissão e o armazenamento de informações sigilosas, o paradigma de criptografia assimétrica trouxe uma nova possibilidade: criar *assinaturas digitais*. Assim como uma assinatura convencional, a assinatura digital tem por finalidade comprovar a procedência de um documento.

Suponha que Alice deseja enviar um documento importante para Bernardo pela Internet, por exemplo uma ordem bancária. Ele precisa ter certeza de que o documento foi enviado por Alice, e não por algum indivíduo malicioso tentando obter ganhos ilícitos. Sabe-se que ambos são usuários de um criptossistema de chave pública, em que Alice é a detentora da chave secreta. A pergunta é: como criar uma assinatura, uma marca única que não deixe dúvidas quanto à procedência do documento enviado por Alice?

A resposta é simples. Alice aplica sua chave secreta sobre o documento, criando uma assinatura. Ao receber o documento assinado, Bernardo aplica a chave pública e confirma a validade da assinatura, concluindo portanto que foi Alice quem de fato enviou o documento, pois somente ela, por ser a única detentora da chave secreta, poderia produzir aquela assinatura.

É fácil produzir assinaturas com o RSA. Suponha que a chave secreta de Alice seja (N, d), e a chave pública (N, e). Ela deseja assinar e enviar a mensagem m. Para isto, ela cria a assinatura $s = m^d \pmod{N}$ e envia o par (m, s) para Bernardo. Ao receber a mensagem, ele aplica a chave pública sobre s, verificando que $s^e \equiv m \pmod{N}$. Assim ele conclui que a mensagem foi enviada por Alice.

Se Eva interceptar a comunicação entre Alice e Bernardo, ela pode tentar modificar a mensagem m, mas não será capaz de gerar uma assinatura válida, pois não conhece a chave secreta. E quando Bernardo descobrir que $s^e \not\equiv m \pmod{N}$, ele saberá que a mensagem foi modificada e a rejeitará.

1.5 A revolução dos *quanta*

Como acabamos de ver, a segurança da criptografia de chave pública depende da intratabilidade de certos problemas matemáticos: a fatoração de inteiros, no caso do RSA, e o cálculo do logaritmo discreto no caso da criptografia baseada em curvas elípticas. Não existem algoritmos clássicos conhecidos que resolvam tais problemas de forma eficiente. No entanto, um novo personagem pode mudar este cenário: o *computador quântico*.

Diferente do computador clássico, cuja unidade fundamental de informação é o *bit*, o computador quântico opera com *qubits* (abreviação para *quantum bits*).

Enquanto um bit só pode assumir um valor, 0 ou 1, de cada vez, um qubit pode literalmente ser 0 *e* 1 ao mesmo tempo, colapsando para um dos dois estados quando é medido.

Somente para dar uma ideia bastante superficial deste conceito, um *qubit* pode ser pensado matematicamente como um vetor unitário em $\mathbb{C}^2$, representado por

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

em que $\alpha, \beta \in \mathbb{C}$ e $|\alpha|^2 + |\beta|^2 = 1$. Os vetores $|0\rangle$ e $|1\rangle$ representam os estados 0 e 1, respectivamente. Portanto, um *qubit* é uma superposição destes dois estados, carregando em si a probabilidade $|\alpha|^2$ de colapsar para o estado $|0\rangle$, e $|\beta|^2$ de colapsar para o estado $|1\rangle$, quando medido. Estamos apenas arranhando a superfície de algo muito profundo. Para uma abordagem matemática completa da computação quântica, recomenda-se a leitura de [10], [11] e [12].

O fato é que, devido aos surpreendentes fenômenos da Mecânica Quântica, os computadores quânticos são capazes de realizar tarefas com muito mais rapidez do que os seus irmãos clássicos. E isto inclui fatorar inteiros e calcular logaritmos discretos, graças ao algoritmo proposto em [13].

É verdade que ainda estamos um pouco longe de um computador quântico de uso geral, que possa ser comprado na loja de eletrodomésticos e levado para casa. Por enquanto, o D-Wave é o único modelo disponível no mercado (Figura 1.5). Ele não é pequeno (ocupa uma área de cerca de $10m^2$), e começar a utilizá-lo não é uma tarefa trivial: a instalação leva quatro semanas, depois das quais ainda são necessárias seis semanas de calibragem, de acordo com informações contidas no site do fabricante[5]. Além disso, o D-Wave não é barato. A Lockheed Martin adquiriu há pouco tempo um D-Wave 1, com processador de 128 *qubits*, por um valor estimado

[5]http://www.dwavesys.com/en/products-services.html

em 10 milhões de dólares. Mais recentemente, a Google anunciou a compra de um D-Wave 2, já com processador de 512 *qubits*. O valor pago pela Google giraria em torno de 15 milhões de dólares!

Segundo os próprios fabricantes, a arquitetura do D-Wave é mais apropriada para resolver problemas de otimização, e não para executar o algoritmo de Shor. Portanto, a princípio, ele não representa uma ameaça aos sistemas de criptografia atuais. No entanto, não é absurdo imaginar que, dentro de algum tempo, décadas talvez, seja desenvolvida uma arquitetura de computador quântico capaz de executar o algoritmo de Shor e, consequentemente, quebrar os sistemas de criptografia atuais.

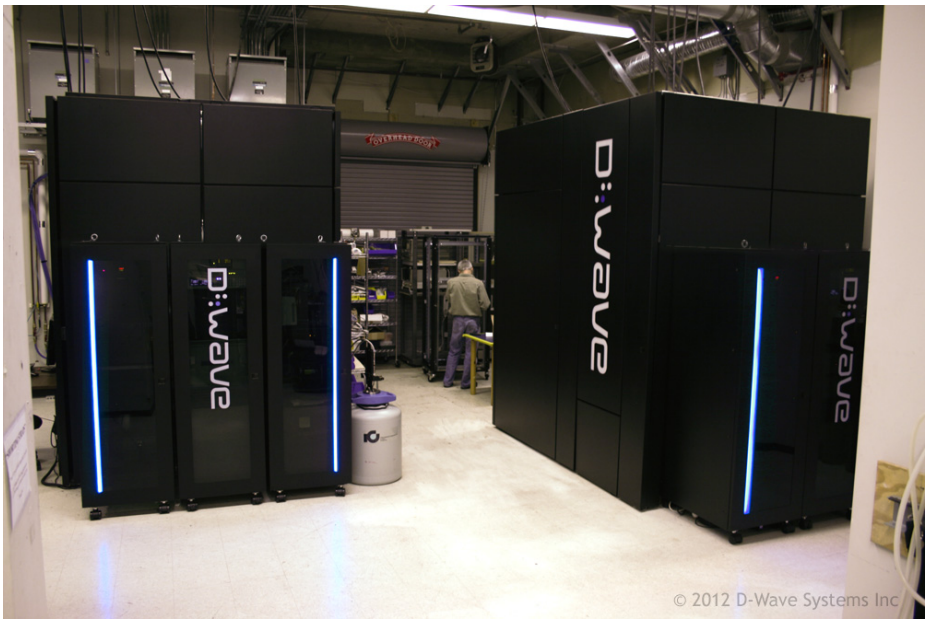

Figura 1.5: DWave, o primeiro computador quântico a ser comercializado. [Fonte: D-Wave Systems].

Diante desta ameaça latente, deve-se pensar agora em criptossistemas alternativos, que poderiam ser utilizados nesta era pós-quântica. Já existe um número razoável de propostas, incluindo sistemas baseados em códigos corretores de erros, funções hash e reticulados. Tais sistemas são baseados em problemas matemáticos que nem mesmo os computadores quânticos, pelo menos até onde se sabe, seriam

capazes de resolver de forma eficiente.

1.6 O que já foi feito...

Os chamados sistemas pós-quânticos, por incrível que pareça, não empregam computação quântica em seu design. São sistemas totalmente clássicos. Ainda assim, permaneceriam de pé, enquanto os sistemas atuais (RSA, curvas elípticas, etc.) sucumbiriam diante de um computador quântico capaz de executar os algoritmos quânticos conhecidos.

O primeiro sistema pós-quântico foi apresentado em [14], na mesma época da criação do RSA. Trata-se de um criptossistema baseado em códigos corretores de erros, cujo problema subjacente é o problema do vetor mais próximo em códigos lineares. Cerca de duas décadas depois foi criado o GGH [15], que possui uma estrutura muito parecida com o sistema McEliece, porém emprega reticulados em vez de códigos lineares.

O GGH não foi o primeiro criptossistema baseado em reticulados. A primeira proposta foi feita por Ajtai e Dwork em [16]. Sua segurança depende da dificuldade computacional de se resolver o pior caso do *problema do único vetor mais curto*. A encriptação é probabilística, sendo efetuada bit por bit. Este detalhe torna inviável a implementação do sistema na prática. Além disso, um ataque ao sistema de Ajtai e Dwork é apresentado em [17]. Algumas modificações neste sistema foram propostas em [18], [19] e [20] e, apesar de sua importância meramente teórica, ele serviu de inspiração para outros sistemas, como em [21], [22] e [23].

Em [24] foi apresentado um ataque ao GGH que explora vulnerabilidades decorrentes da escolha dos parâmetros propostos originalmente. Apesar de tais

vulnerabilidades serem corrigíveis com a mudança de parâmetros, isto tornaria o sistema inviável segundo o autor, porque o tamanho das chaves excederia os limites aceitáveis. Tudo isto fez com que o GGH fosse declarado oficialmente *morto* [25].

O único sistema baseado em reticulados que parece resistir a todos os esforços criptoanalíticos é o NTRU [26], que na verdade é uma variante do GGH que utiliza classes especiais de reticulados. Além disso, diferentemente do GGH, grandes dimensões não tornam o NTRU inviável.

Mais recentemente, Yoshino e Kunihiro propuseram modificações que supostamente aumentam a segurança do GGH [27], além de tornar determinístico o processo de decriptação, diferentemente do sistema original, no qual a decriptação é suscetível a erros.

1.7 ...e o que há por fazer

Este trabalho apresentará uma análise completa da versão do GGH proposta por Yoshino e Kunihiro, que será batizada de GGH-YK. O objetivo é mostrar que, na prática, o GGH-YK não se comporta de maneira diferente do GGH original. Também serão propostas modificações no GGH-YK, a fim de torná-lo efetivamente diferente do GGH. Serão discutidos ainda esquemas de autenticação, baseados em uma variante do GGH-YK, explorando o conceito de protocolos de autenticação por *challenge-response*.

O restante deste trabalho está organizado da seguinte maneira: o Capítulo 2 será destinado a uma breve revisão de Álgebra Linear. No Capítulo 3 será estudada a teoria de reticulados, desde os conceitos fundamentais até os algoritmos de redução de bases. O sistema GGH e sua variante GGH-YK serão apresentados no Capítulo

4. No Capítulo 5 serão propostas modificações no GGH-YK, dando origem a uma variante que batizaremos de GGHYK-M. Propostas de protocolos de autenticação baseados no GGHYK-M serão apresentadas no Capítulo 6. Finalmente, no Capítulo 7 serão apresentados resultados de testes e simulações computacionais.

2 BREVE REVISÃO DE ÁLGEBRA LINEAR

> "Povoamos sonhos, somos sombras errando através de florestas impossíveis, em que as árvores são casas, costumes, ideias, ideais e filosofias." (Fernando Pessoa)

Este capítulo tem por objetivo relembrar alguns conceitos básicos de Álgebra Linear, que serão mencionados ao longo deste trabalho. Não será feita uma abordagem exaustiva e minuciosa de todos os tópicos. Para mais detalhes, pode-se consultar [28] ou [29].

2.1 Espaços vetoriais

Um espaço vetorial sobre um corpo F é um conjunto V, cujos elementos são denominados *vetores*, munido de duas operações: a *soma vetorial* e a *multiplicação por escalar*, em que *escalar* é um elemento do corpo F. Estas operações devem satisfazer as seguintes propriedades, para todos os vetores[1] $\mathbf{u}, \mathbf{v}, \mathbf{w} \in V$ e escalares $a, b \in F$.

1. Associatividade da soma: $\mathbf{u} + (\mathbf{v} + \mathbf{w}) = (\mathbf{u} + \mathbf{v}) + \mathbf{w}$.
2. Comutatividade da soma: $\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}$.
3. Elemento neutro da soma: existe um elemento $\mathbf{0} \in V$, chamado de *vetor nulo*, tal que $\mathbf{v} + \mathbf{0} = \mathbf{v}$, para todo $\mathbf{v} \in V$.

[1] Para que não haja ambiguidade, vetores serão sempre representados por letras minúsculas em negrito.

4. Inverso aditivo: para todo $\mathbf{v} \in V$, existe um *inverso aditivo*, denotado por $-\mathbf{v}$, tal que $\mathbf{v} + (-\mathbf{v}) = \mathbf{0}$.

5. Distributividade: $a(\mathbf{u} + \mathbf{v}) = a\mathbf{u} + a\mathbf{v}$ e $(a + b)\mathbf{u} = a\mathbf{u} + b\mathbf{u}$.

6. Associatividade da multiplicação por escalar: $a(b\mathbf{u}) = (ab)\mathbf{u}$.

7. Elemento neutro da multiplicação por escalar: $1\mathbf{u} = \mathbf{u}$, em que 1 é o elemento neutro da multiplicação em F.

Se $W \subset V$ satisfaz as propriedades acima, dizemos que W é um *subespaço* vetorial de V. Quando V é o conjunto de n-uplas formadas por elementos de F, dizemos que ele é um *espaço de coordenadas*, denotado por F^n. O exemplo mais comum de espaço de coordenadas é $\mathbb{R}^n$, denominado *espaço euclidiano* ou *espaço cartesiano real* de dimensão n, que é definido sobre o corpo dos reais:

$$\mathbb{R}^n = \left\{ (a_1, \cdots, a_n)^T \mid a_i \in \mathbb{R}, \text{ para } i = 1, \cdots, n \right\}. \tag{2.1}$$

O motivo de usarmos a transposição em (2.1) é que, por convenção, consideraremos os elementos de $\mathbb{R}^n$ como *vetores coluna*.

2.1.1 Normas e produtos internos

O espaço cartesiano n-dimensional é munido, além das operações descritas anteriormente, de duas funções especiais: a *norma* e o *produto interno*. Como será discutido mais à frente, tais funções permitem estabelecer as noções de tamanho, distância e ângulo.

Definição 2.1.1 *Uma norma em $\mathbb{R}^n$ é uma função $||.|| : \mathbb{R}^n \longrightarrow \mathbb{R}$, satisfazendo as seguintes propriedades para todos $a \in \mathbb{R}$ e $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$:*

1. $||\mathbf{u}|| \geq 0$;

2. $||\mathbf{u}|| = 0 \Leftrightarrow \mathbf{u} = 0$;

3. $||a\mathbf{u}|| = |a|||\mathbf{u}||$;

4. $||\mathbf{u} + \mathbf{v}|| \leq ||\mathbf{u}|| + ||\mathbf{v}||$.

Um vetor $\mathbf{u} \in \mathbb{R}^n$ é dito *unitário* se $||\mathbf{u}|| = 1$. A norma mais comum e intuitiva é a chamada norma l_2, também denominada *norma euclidiana*, definida da seguinte maneira:

$$||(a_1, \cdots, a_n)^T||_2 = \sqrt{a_1^2 + \cdots + a_n^2}. \tag{2.2}$$

Esta norma transmite a ideia convencional de *tamanho*, mas vale ressaltar que esta noção não é a única. De modo geral, define-se a norma l_p, para $p = 1, 2, \cdots$, dada por

$$||(a_1, \cdots, a_n)^T||_p = \left(\sum_{i=1}^{n} |a_i|^p \right)^{\frac{1}{p}}, \tag{2.3}$$

e a norma l_∞

$$||(a_1, \cdots, a_n)^T||_\infty = \max \{|a_1|, \cdots, |a_n|\}. \tag{2.4}$$

Ao longo deste trabalho, a notação $||\mathbf{u}||$ deixará implícito que se trata da norma l_2 do vetor $\mathbf{u}$. Quando outra norma for utilizada, isto ficará explícito.

Observe que uma norma induz naturalmente uma noção de *distância*. De fato, a distância entre $\mathbf{u}$ e $\mathbf{v}$ é dada por $||\mathbf{u} - \mathbf{v}||$. De maneira mais formal, as funções que medem distância são denominadas *métricas*. Assim, podemos concluir que toda norma induz uma métrica.

Definição 2.1.2 *Um produto interno em* $\mathbb{R}^n$ *é uma função* $\langle ., . \rangle : \mathbb{R}^n \times \mathbb{R}^n \longrightarrow \mathbb{R}$ *que satisfaz as seguintes propriedades, para todos* $a, b \in \mathbb{R}$ *e* $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathbb{R}^n$*:*

1. *Bilinearidade:* $\langle a\mathbf{u} + b\mathbf{v}, \mathbf{w}\rangle = a\langle \mathbf{u}, \mathbf{w}\rangle + b\langle \mathbf{v}, \mathbf{w}\rangle$*;*

2. *Simetria:* $\langle \mathbf{u}, \mathbf{v}\rangle = \langle \mathbf{v}, \mathbf{u}\rangle$*;*

3. *Positividade:* $\langle \mathbf{u}, \mathbf{u}\rangle \geq 0$*, e* $\langle \mathbf{u}, \mathbf{u}\rangle = 0 \Leftrightarrow u = 0$.

O produto interno usual entre dois vetores de $\mathbb{R}^n$ é definido da seguinte maneira:

$$\langle \mathbf{u}, \mathbf{v}\rangle = \sum_{i=1}^{n} u_i v_i. \tag{2.5}$$

Neste trabalho faremos uso exclusivamente do produto interno definido acima, mas vale ressaltar que, assim como ocorre com a norma, a definição de produto interno não é única. Observe que o produto interno usual entre os vetores $\mathbf{u}$ e $\mathbf{v}$ é dado por $\mathbf{u}^T\mathbf{v}$.

Da mesma maneira que uma norma induz uma noção de distância, todo produto interno induz uma norma. De fato, é fácil observar que

$$||\mathbf{u}|| = \sqrt{\langle \mathbf{u}, \mathbf{u}\rangle} \tag{2.6}$$

satisfaz todas as propriedades da Definição 2.1.1. Em particular, o produto interno usual induz a norma euclidiana.

O produto interno entre dois vetores nos dá uma noção do *ângulo* entre eles, servindo como critério de ortogonalidade. Dizemos que dois vetores $\mathbf{u}$ e $\mathbf{v}$ são *ortogonais entre si* se $\langle \mathbf{u}, \mathbf{v}\rangle = 0$. Um conjunto $S = \{\mathbf{u}_1, \cdots, \mathbf{u}_k\}$ é dito *ortogonal* se $\langle \mathbf{u}_i, \mathbf{u}_j\rangle = 0$, para todo $i \neq j$. Se, além disso, todos os vetores de S forem unitários, o conjunto é dito *ortonormal*.

Se V é um subespaço vetorial de $\mathbb{R}^n$, o seu *complemento ortogonal* é o espaço $V^\perp$ formado pelos vetores que são ortogonais a todos os vetores de V, ou seja,

$$V^\perp = \{\mathbf{u} \in \mathbb{R}^n \mid \langle \mathbf{u}, \mathbf{v}\rangle = 0,\ \forall\, \mathbf{v} \in V\}. \tag{2.7}$$

2.1.2 Bases

Dado um conjunto de vetores $\{\mathbf{u}_1, \cdots, \mathbf{u}_n\}$, uma *combinação linear* deles é o vetor

$$a_1\mathbf{u}_1 + \cdots + a_n\mathbf{u}_n,$$

em que $a_1, \cdots, a_n \in \mathbb{R}$. Uma combinação linear é dita *trivial* se $a_i = 0$, para todo $i = 1, \cdots, n$. Caso contrário, a combinação linear é dita *não-trivial*.

Um conjunto de vetores é dito *linearmente independente* (L.I.) se o vetor nulo só pode ser expresso como combinação linear trivial desses vetores. Uma *base* de um espaço vetorial é um conjunto L.I. de vetores que *gera* todo o espaço, ou seja, todo vetor do espaço pode ser escrito como combinação linear dos vetores da base. O espaço gerado pelos vetores $\mathbf{u}_1, \cdots, \mathbf{u}_k$ será denotado por $\text{span}(\mathbf{u}_1, \cdots, \mathbf{u}_k)$, e o complemento ortogonal deste espaço por $\text{span}(\mathbf{u}_1, \cdots, \mathbf{u}_k)^\perp$.

É comum agruparmos os vetores de uma base em uma estrutura matricial. Assim, a base $\{\mathbf{u}_1, \cdots, \mathbf{u}_n\}$ é associada a uma matriz

$$U = \begin{pmatrix} | & | & \cdots & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_n \\ | & | & \cdots & | \end{pmatrix}$$

que por uma questão de simplicidade será escrita na notação compacta

$$U = [\mathbf{u}_1 \cdots \mathbf{u}_n].$$

Uma base será sempre representada por uma letra maiúscula, e seus vetores pela *mesma letra* minúscula, em negrito. Assim, não haverá dúvidas de que os vetores de uma base W, por exemplo, são $\mathbf{w}_1, \cdots, \mathbf{w}_n$.

2.1.3 Bases Ortogonais e Ortonormais

Uma base B é dita *ortogonal* ou *ortonormal* se os vetores da base $\{\mathbf{b}_1, \cdots, \mathbf{b}_n\}$ formam um conjunto ortogonal ou ortonormal, respectivamente. Um exemplo de base ortonormal é a *base canônica* de $\mathbb{R}^n$, dada pela matriz $E = [\mathbf{e}_1 \cdots \mathbf{e}_n]$, em que

$$\mathbf{e}_k = (a_1, \cdots, a_n)^T, \text{ tal que } a_i = \begin{cases} 1, \text{ se } i = k; \\ 0, \text{ caso contrário.} \end{cases}$$

Como veremos um pouco mais adiante, a partir de qualquer base é possível obter uma base ortogonal (ou ortonormal) de $\mathbb{R}^n$. Antes disso, convém relembrar o conceito de projeção ortogonal.

Sejam $\mathbf{u}$ e $\mathbf{v}$ dois vetores de $\mathbb{R}^n$. A projeção ortogonal de $\mathbf{u}$ sobre $\mathbf{v}$ é o vetor dado por

$$\mathbf{p}^{\mathbf{u}}_{\mathbf{v}} = \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{||\mathbf{v}||^2} \mathbf{v}. \tag{2.8}$$

Se $\mathbf{v}_1, \cdots, \mathbf{v}_k$ são vetores linearmente independentes, a projeção ortogonal de $\mathbf{u}$ sobre $\text{span}(\mathbf{v}_1, \cdots, \mathbf{v}_k)$ é dada por

$$\mathbf{p}^{\mathbf{u}}_{\text{span}(\mathbf{v}_1, \cdots, \mathbf{v}_k)} = \sum_{j=1}^{k} \frac{\langle \mathbf{u}, \mathbf{v}_j \rangle}{||\mathbf{v}_j||^2} \mathbf{v}_j. \tag{2.9}$$

É fácil ver então que a projeção de $\mathbf{u}$ sobre $\text{span}(\mathbf{v}_1, \cdots, \mathbf{v}_k)^{\perp}$ é dada por

$$\mathbf{p}^{\mathbf{u}}_{\text{span}(\mathbf{v}_1, \cdots, \mathbf{v}_k)^{\perp}} = \mathbf{u} - \sum_{j=1}^{k} \frac{\langle \mathbf{u}, \mathbf{v}_j \rangle}{||\mathbf{v}_j||^2} \mathbf{v}_j. \tag{2.10}$$

Todo espaço vetorial de dimensão finita, munido de produto interno, admite uma base ortogonal, que pode ser obtida através do processo de Gram-Schmidt. Dada uma base arbitrária B, a ideia é obter uma base ortogonal B^* tal que

$$\text{span}(\mathbf{b}_1^*, \cdots, \mathbf{b}_k^*) = \text{span}(\mathbf{b}_1, \cdots, \mathbf{b}_k),$$

para todo $k = 1, \cdots, n$. Dizemos que B^* é a *base de Gram-Schmidt associada a* B. A Figura 2.1 dá uma ideia bidimensional do processo de Gram-Schmidt.

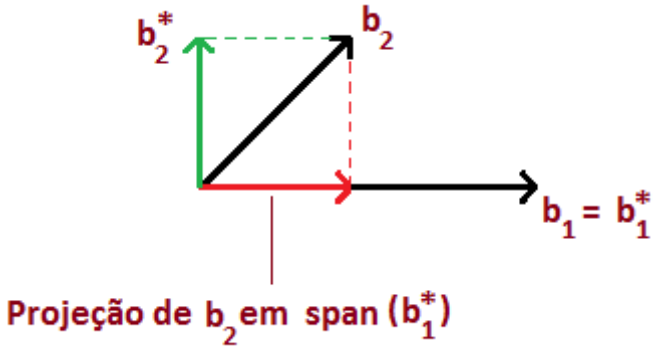

Figura 2.1: Ortogonalização de Gram-Schmidt em duas dimensões.

Para obter B^*, faz-se $\mathbf{b}_1^* = \mathbf{b}_1$ e, para cada $j = 2, \cdots, n$, projeta-se $\mathbf{b}_j$ em $\text{span}(\mathbf{b}_1, \cdots, \mathbf{b}_{j-1})^\perp$, de acordo com o Algoritmo 1.

Algoritmo 1: Gram-Schmidt

Entrada: Uma base B.
Saída: Uma base ortogonal B^*.
início
  $\mathbf{b}_1^* \leftarrow \mathbf{b}_1$;
  para $j = 2, \cdots, n$ **faça**
    $\mathbf{b}_j^* \leftarrow \mathbf{b}_j$;
    para $i = 1, \cdots, j-1$ **faça**
      $\mu \leftarrow \langle \mathbf{b}_i^*, \mathbf{b}_j \rangle / \langle \mathbf{b}_i^*, \mathbf{b}_i^* \rangle$;
      $\mathbf{b}_j^* \leftarrow \mathbf{b}_j^* - \mu \mathbf{b}_i^*$
    fim para
  fim para
fim

Observe que a base B^* se relaciona com B pela matriz triangular superior M, cujas entradas $\mu_{i,j}$, denominadas *coeficientes de Gram-Schmidt*, são dadas por

$$\mu_{i,j} = \begin{cases} \dfrac{\langle \mathbf{b}_i^*, \mathbf{b}_j \rangle}{\langle \mathbf{b}_i^*, \mathbf{b}_i^* \rangle}, & \text{se } i \leq j; \\ 0, & \text{caso contrário.} \end{cases}$$

De fato, é fácil verificar que $B = B^* M$. Assim temos, para todo $j = 1, \cdots, n$,

$$\mathbf{b}_j = \sum_{i=1}^{j} \mu_{i,j} \mathbf{b}_i^*.$$

Em outras palavras, $\mu_{i,j}$ é o coeficiente de $\mathbf{b}_i^*$ em $\mathbf{b}_j$. Pode-se dizer então que $\mathbf{b}_j^*$ é

o componente de $\mathbf{b}_j$ ortogonal a $\text{span}(\mathbf{b}_1, \cdots, \mathbf{b}_{j-1})$. Além disso, pode-se constatar que $\mu_{j,j} = 1$, para todo $j = 1, \cdots, n$.

2.2 Autovetores e Autovalores

Uma matriz pode ser pensada como um *operador linear* que transforma os vetores de um subespaço em vetores de outro (ou do mesmo) subespaço. Por exemplo, a matriz

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

projeta todo vetor de $\mathbb{R}^2$ no eixo das abscissas. Em algumas situações, um vetor é transformado em um múltiplo de si mesmo. Neste caso, dizemos que se trata de um *autovetor* ou *vetor característico*.

Definição 2.2.1 *Seja T um operador linear (uma matriz quadrada $n \times n$) com entradas reais. Um vetor $\mathbf{0} \neq \mathbf{v} \in \mathbb{R}^n$ é dito um autovetor (ou vetor característico) de T se existe $\lambda \in \mathbb{R}$ tal que*

$$T\mathbf{v} = \lambda\mathbf{v}. \tag{2.11}$$

O número real λ é dito o *autovalor* ou *valor característico* correspondente ao autovetor $\mathbf{v}$. O espaço formado por todos os autovetores associados a um determinado autovalor, juntamente com o vetor $\mathbf{0}$, é chamado de *autoespaço* ou *espaço característico* correspondente àquele autovalor. Observe que, da equação (2.11), temos que $(T - \lambda I)\mathbf{v} = \mathbf{0}$, com $\mathbf{v} \neq \mathbf{0}$, o que implica em

$$\det(T - \lambda I) = 0. \tag{2.12}$$

Os autovalores de T são as soluções (se existirem) da equação (2.12). O *raio espectral* de T é o máximo dos módulos dos autovalores de T, ou seja,

$$\rho(T) = \max\{|\lambda|, \text{ em que } \lambda \text{ é autovalor de } T\}. \tag{2.13}$$

No próximo capítulo será feito um estudo dos reticulados, que podem ser vistos como *análogos discretos* dos espaços vetoriais.

3 RETICULADOS

> "De modo que desligo de mim como aos dois braços de um amplexo, os dois grandes tédios que me apertam — o tédio de poder viver só o Real, e o tédio de poder conceber só o Possível." (Fernando Pessoa)

Neste capítulo será feito um estudo da teoria de reticulados, com uma descrição dos conceitos de bases e domínios fundamentais, dos problemas computacionais relacionados e também dos chamados *algoritmos de redução*.

Vale ressaltar que esta é uma área da Matemática rica em aplicações, não apenas na criptografia, mas também em processamento de sinais, como estimativas de frequência e espaço de cores em imagens JPEG, e detecção e pré-codificação de dados em sistemas de comunicação wireless (veja [30] para mais detalhes).

Do ponto de vista teórico, reticulados são objetos de estudo da *geometria de números*. Para um tratado completo do assunto, recomenda-se a leitura de [31].

3.1 Bases de um Reticulado

Reticulados são *subgrupos aditivos discretos de* $\mathbb{R}^n$, ou equivalentemente, $\mathbb{Z}$-*módulos livres finitamente gerados*. Um conjunto $L \subset \mathbb{R}^n$ é dito *discreto* se existe $\gamma > 0$ tal que, para todos $x \neq y \in L$, temos $||x - y|| \geq \gamma$. Portanto, reticulados são fechados em relação à soma (são subgrupos aditivos) e são discretos, isto é, existe uma cota inferior para a distância entre quaisquer dois de seus elementos.

Assim como os espaços vetoriais, os reticulados podem ser descritos através de bases de vetores, porém, como veremos a seguir, somente combinações lineares com coeficientes inteiros são permitidas.

Definição 3.1 *Seja* $B = [\mathbf{b}_1 \cdots \mathbf{b}_m] \in \mathbb{R}^{n \times m}$*, em que* $\mathbf{b}_1, \cdots, \mathbf{b}_m$ *são vetores linearmente independentes de* $\mathbb{R}^n$*. O reticulado gerado por* B *é o conjunto*

$$L(B) = \left\{ \sum_{i=1}^{n} a_i \mathbf{b}_i,\ a_i \in \mathbb{Z},\ \textit{para } i = 1, \cdots, n \right\}.$$

A matriz B é dita a *base* de $L(B)$, m é o seu posto e n a sua dimensão. Em particular, se $m = n$, dizemos que o reticulado possui *posto completo*. Ao longo deste trabalho, faremos referência apenas a reticulados de posto completo.

Se $L(B)$ é um reticulado de posto completo n, então ele é a imagem da transformação linear descrita por B, cujo domínio é $\mathbb{Z}^n$. Ou seja,

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

Observe que, pela Definição 3.1, as entradas dos vetores de um reticulado não são necessariamente inteiras. No caso particular em que isto ocorre, dizemos que o reticulado é *inteiro*. Por exemplo, $\mathbb{Z}^n$ é um reticulado inteiro.

Por definição, todo vetor de $L(B)$ é uma combinação linear com coeficientes inteiros dos vetores de B. Se outra base $W = [\mathbf{w}_1 \cdots \mathbf{w}_n]$ gera o mesmo reticulado, ou seja, se $L(W) = L(B)$, então os vetores de W também são combinações lineares dos vetores de B, isto é,

$$\begin{cases} \mathbf{w}_1 &= u_{1_1}\mathbf{b}_1 + u_{2_1}\mathbf{b}_2 + \cdots + u_{n_1}\mathbf{b}_n \\ \mathbf{w}_2 &= u_{1_2}\mathbf{b}_1 + u_{2_2}\mathbf{b}_2 + \cdots + u_{n_2}\mathbf{b}_n \\ \vdots \quad \vdots & \qquad\qquad \vdots \\ \mathbf{w}_n &= u_{1_n}\mathbf{b}_1 + u_{2_n}\mathbf{b}_2 + \cdots + u_{n_n}\mathbf{b}_n \end{cases}$$

em que $u_{ij} \in \mathbb{Z}$. As equações acima podem ser reescritas matricialmente como

$$[\mathbf{w}_1 \cdots \mathbf{w}_n] = [\mathbf{b}_1 \cdots \mathbf{b}_n][\mathbf{u}_1 \cdots \mathbf{u}_n],$$

ou seja, $W = BU$, em que as entradas de U são inteiras. Para expressar os vetores de B como combinações lineares dos vetores de W, temos a equação matricial $B = WU^{-1}$. Então, os coeficientes de U^{-1} também devem ser inteiros. Como $\det(U)\det(U^{-1}) = 1$, concluímos que $\det(U) = \pm 1$.

A matriz U é dita *unimodular*, e quaisquer duas bases de um reticulado se relacionam por uma matriz unimodular. Em outras palavras, **se A e B são duas bases para o mesmo reticulado, então existe uma matriz unimodular U tal que $A = BU$.**

3.1.1 Domínios Fundamentais

Seja B uma base. Um *domínio fundamental* de $L(B)$ é a imagem do cubo semi-aberto $\mathcal{C}_n = [0, 1)^n$ por B, ou seja,

$$\mathcal{F}_B = B\mathcal{C}_n = \{t_1\mathbf{b}_1 + \cdots + t_n\mathbf{b}_n;\ 0 \leq t_i < 1\}.$$

Portanto, um domínio fundamental é um paralelepípedo n-dimensional formado pelos vetores da base (Figura 3.1), e seu volume é dito o *determinante* (ou *discriminante*) de $L(B)$, denotado por $\det(L(B))$. A proposição a seguir relaciona o determinante do reticulado com o determinante da base:

Proposição 3.1 *Seja $B = [\mathbf{b}_1 \cdots \mathbf{b}_n]$ uma base. Então,*

$$\det(L(B)) = |\det(B)|.$$

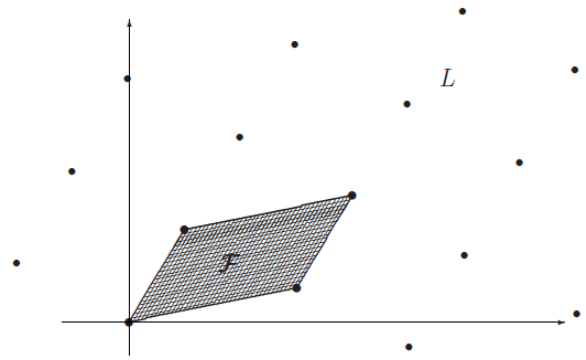

Figura 3.1: Domínio fundamental de um reticulado em duas dimensões.

Demonstração: seja $\mathcal{F}_B$ um domínio fundamental de $L(B)$. Então,

$$\text{Vol}(\mathcal{F}_B) = \int_{\mathcal{F}_B} dx_1 dx_2 \cdots dx_n.$$

Aplicando a mudança de variáveis $(x_1, \cdots, x_n) = t_1\mathbf{b}_1 + \cdots + t_n\mathbf{b}_n$, temos a equação matricial $\mathbf{x} = B\mathbf{t}$. Usando o fato de que $\mathcal{F}_B$ é a imagem de $\mathcal{C}_n$ por B, temos

$$\text{Vol}(\mathcal{F}_B) = \int_{B\mathcal{C}_n} dx_1 \cdots dx_n = \int_{\mathcal{C}_n} |\det(B)| dt_1 \cdots dt_n =$$
$$= |\det(B)|\text{Vol}(\mathcal{C}_n) = |\det(B)|.$$

Portanto, $\det(L(B)) = |\det(B)|$. □

Como vimos anteriormente, quaisquer duas bases se relacionam através de uma matriz cujo determinante é ± 1. Logo, **o determinante do reticulado é invariante em relação a mudanças de base.**

Como a transposição de matrizes preserva o módulo do determinante, temos que $\det(L(B)) = \sqrt{\det(B^T B)}$. Portanto, o determinante de $L(B)$ é dado por

$$\det(L(B)) = (\det(G_B))^{1/2}, \tag{3.1}$$

em que $G_B = B^T B = [\langle \mathbf{b}_i, \mathbf{b}_j \rangle_{1 \leq i,j \leq n}]$ é a *matriz de Gram* associada à base B. Pode-se observar que toda base induz um particionamento do reticulado em cópias

do domínio fundamental, ou seja, em paralelepípedos cujos vértices são os pontos do reticulado. Bases diferentes induzem particionamentos diferentes, mas os domínios fundamentais em todos eles possuem o mesmo volume (veja a Figura 3.2).

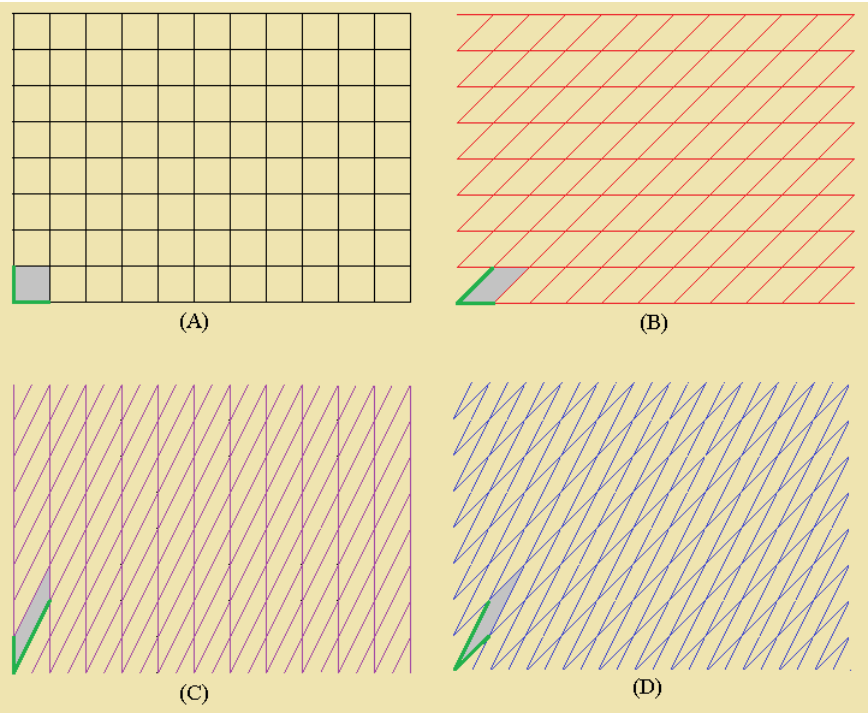

Figura 3.2: Particionamentos induzidos por quatro bases de um mesmo reticulado. A área do domínio fundamental de todos os particionamentos é constante, graças à invariância do determinante do reticulado.

A seguir veremos que existe uma cota superior para o volume de um domínio fundamental, que depende essencialmente dos vetores da base. O próximo resultado, conhecido como *desigualdade de Hadamard*, generaliza o seguinte fato geométrico: a área (ou volume) de um paralelogramo (ou paralelepípedo) é menor ou igual ao produto das medidas de seus lados. Este fato, bastante simples de enxergar em dimensões 2 e 3, vale então para dimensões arbitrárias.

Proposição 3.2 (Hadamard) *Seja B uma base. Então*

$$\det(L(B)) \leq \prod_{j=1}^{n} ||\mathbf{b}_j||, \tag{3.2}$$

valendo a igualdade se, e somente se B for ortogonal.

Demonstração: vamos definir uma base ortonormal B', de modo similar ao que fizemos no Cap. 2, fazendo $\mathbf{b}'_1 = \frac{1}{||\mathbf{b}_1||}\mathbf{b}_1$ e, para cada $j = 2, \cdots, n$,

$$\mathbf{b}'_j = \frac{1}{||\mathbf{b}^*_j||}\mathbf{b}^*_j.$$

Assim temos

$$\mathbf{b}_j = \sum_{i=1}^{j} \mu'_{i,j}\mathbf{b}'_i,$$

em que $\mu'_{i,j} = \langle \mathbf{b}'_i, \mathbf{b}_j \rangle$, para todo $i < j$. Note que $B = B'M'$, em que M' é a matriz triangular superior dos coeficientes $\mu'_{i,j}$. Portanto, temos que

$$\begin{aligned}
\det^2(B) &= \det(B^T B) \\
&= \det(M'^T (B')^T (B') M') \\
&= \det(M'^T M') \\
&= \det^2(M') \\
&= \prod_{j=1}^{n} \left(\mu'_{j,j}\right)^2 \\
&\leq \prod_{j=1}^{n} \sum_{i=1}^{j} \left(\mu'_{i,j}\right)^2 \\
&= \prod_{j=1}^{n} ||\mathbf{b}_j||^2
\end{aligned}$$

Em particular, a igualdade em (3.2) é válida se, e somente se

$$\left(\mu'_{j,j}\right)^2 = \sum_{i=1}^{j} \left(\mu'_{i,j}\right)^2,$$

para cada j, ou seja, se

$$\sum_{i=1}^{j-1}(\mu'_{i,j})^2 = 0$$

para todo j. Portanto, a igualdade é válida se, e somente se a base B for ortogonal. □

É fácil notar que, quanto mais a base B estiver próxima de ser ortogonal, mais o lado esquerdo da desigualdade (3.2) se aproxima do lado direito. Pode-se medir então o quanto uma base B está próxima de ser ortogonal através de seu *defeito ortogonal*, dado por

$$\mathcal{D}(B) = \frac{||\mathbf{b}_1|| \cdots ||\mathbf{b}_n||}{|\det(B)|},$$

ou ainda pela sua *Razão de Hadamard*, que é dada por

$$\mathcal{H}(B) = \left(\frac{1}{\mathcal{D}(B)}\right)^{\frac{1}{n}}.$$

Por (3.2), temos $\mathcal{D}(B) \geq 1$, enquanto $0 < \mathcal{H}(B) \leq 1$. Se B for ortogonal, então $\mathcal{D}(B) = \mathcal{H}(B) = 1$. À medida que a base se torna menos ortogonal, seu defeito ortogonal aumenta indefinidamente, enquanto sua Razão de Hadamard se aproxima de 0.

Para fins criptográficos, o defeito ortogonal (ou a Razão de Hadamard) serve como medida da *qualidade* de uma base. Dizemos então que uma base B será tanto melhor quanto mais próximo de 1 estiver seu defeito ortogonal/Razão de Hadamard. É possível gerar eficientemente bases muito boas e, a partir delas, bases muito ruins. Porém, como será discutido na próxima seção, achar uma base boa a partir de uma base ruim é um problema computacionalmente difícil.

3.2 Problemas Computacionais

Existem dois problemas fundamentais na teoria de reticulados: o *problema do vetor mais curto* (*Shortest Vector Problem* ou SVP) e o *problema do vetor mais próximo* (*Closest Vector Problem* ou CVP). O primeiro consiste em, dado um reticulado $L(B)$, encontrar um vetor $\mathbf{v} \in L(B)$ tal que $||\mathbf{v}||$ seja mínima. Já o segundo consiste em, dados um reticulado $L(B)$ e um vetor $\mathbf{c} \notin L(B)$, encontrar $\mathbf{v} \in L(B)$ tal que $||\mathbf{c} - \mathbf{v}||$ seja mínima.

Tanto o SVP quanto o CVP são NP-difíceis de acordo com [32] e [33], mas existem algoritmos que permitem resolvê-los de forma aproximada.

O CVP pode ser resolvido pelo algoritmo proposto em [34], desde que se conheça uma base boa B (com razão de Hadamard não muito pequena). A ideia do algoritmo é a seguinte: dados um reticulado $L(B)$ e um vetor $\mathbf{w} \notin L(B)$, calculam-se as coordenadas de $\mathbf{w}$ na base B. Essas coordenadas são arredondadas para o inteiro mais próximo, e obtém-se o vetor do reticulado correspondente às coordenadas arredondadas. Quanto melhor for a base, melhor será a solução encontrada.

Graficamente, o algoritmo de Babai consiste em enxergar o ponto $\mathbf{w}$ dentro de um domínio fundamental, de modo que o ponto mais próximo corresponda a um dos vértices desse domínio. Tenta-se então localizar este vértice. Se a base for muito ruim, pode ser que o ponto mais próximo de $\mathbf{w}$ não corresponda a um vértice do domínio. Neste caso, a resposta dada pelo algoritmo pode estar muito distante do ponto mais próximo.

Pode-se ainda reduzir o CVP ao SVP, através da técnica de *imersão* apresentada em [35], em que se tenta encontrar o vetor mais curto em um reticulado de dimensão maior, para então resolver o CVP no reticulado original. Mais precisa-

mente, dada uma base B e um vetor $\mathbf{c}$, desejamos encontrar o vetor $\mathbf{v} \in L(B)$ mais próximo de $\mathbf{c}$. A ideia é construir o reticulado gerado pela matriz

$$W = \begin{pmatrix} \mathbf{b}_1 & \cdots & \mathbf{b}_n & \mathbf{c} \\ 0 & \cdots & 0 & 1 \end{pmatrix}.$$

e encontrar o vetor mais curto em $L(W)$. Se $(\mathbf{r}^T, 1)^T$ é o vetor mais curto em $L(W)$, então $\mathbf{v} = \mathbf{c} - \mathbf{r}$ é o vetor de $L(B)$ mais próximo de $\mathbf{c}$. Observe que, de fato, $\mathbf{v} \in L(B)$, pois a coordenada de $(\mathbf{r}^T, 1)^T$ em $(\mathbf{c}^T, 1)^T$ é 1 (já que todos os outros vetores de W têm a última coordenada nula). Assim, $\mathbf{c} - \mathbf{r}$ é uma combinação linear dos vetores $\mathbf{b}_i$, ou seja, $\mathbf{c} - \mathbf{r} \in L(B)$.

Suponha então que $(\mathbf{r}^T, 1)^T$ seja o vetor mais curto em $L(W)$, mas $\mathbf{v} = \mathbf{c} - \mathbf{r}$ não seja o vetor mais próximo de $\mathbf{c}$. Então, existem vetores $\mathbf{w} \in L(B)$ e $\mathbf{s} \notin L(B)$ tais que $\mathbf{c} = \mathbf{w} + \mathbf{s}$, e $||\mathbf{c} - \mathbf{w}|| = ||\mathbf{s}|| < ||\mathbf{r}||$. Pela construção que acabamos de fazer, conclui-se que $(\mathbf{s}^T, 1)^T \in L(W)$. Mas então $||(\mathbf{s}^T, 1)^T|| < ||(\mathbf{r}^T, 1)^T||$, contrariando a hipótese de que $(\mathbf{r}^T, 1)^T$ é o vetor mais curto em $L(W)$.

O problema do vetor mais curto também pode ser resolvido de forma aproximada através dos chamados algoritmos de *redução de base*, que serão discutidos na Seção 3.3. Espera-se que um dos vetores da base encontrada seja uma aproximação para o vetor mais curto do reticulado. Em dimensão 2, o algoritmo de redução de Gauss resolve o problema de forma exata. Trata-se de uma adaptação do método de ortogonalização de Gram-Schmidt, em que os coeficientes são arredondados para garantir que todas as combinações lineares envolvidas sejam inteiras.

Em dimensões maiores, existem algoritmos como o LLL, proposto em [36], e o BKZ, apresentado em [37].

3.3 Bases Reduzidas

A *teoria da redução* é o estudo de técnicas para obter bases com certas propriedades desejáveis, como vetores próximos de serem ortogonais e com norma relativamente pequena. Uma base é dita reduzida quando satisfaz tais propriedades, e os algoritmos usados para obter uma base reduzida a partir de uma base qualquer são denominados *algoritmos de redução*. O problema da redução de base é considerado difícil em dimensões arbitrárias e está diretamente relacionado à redução de formas quadráticas positivas-definidas, que já havia sido estudada em [38] e [39].

3.3.1 Redução em Duas Dimensões

Iniciamos a nossa discussão com o problema da redução de base em dimensão 2. Seja então B uma base de um reticulado bidimensional que gostaríamos de reduzir. Seria muito conveniente poder aplicar a ortogonalização de Gram-Schmidt, projetando $\mathbf{b}_2$ sobre o complemento ortogonal do espaço gerado por $\mathbf{b}_1$. Infelizmente, isto não é possível de modo geral, pois seria equivalente a substituir $\mathbf{b}_2$ por uma combinação linear cujos coeficientes não são necessariamente inteiros, mais especificamente o coeficiente de Gram-Schmidt $\mu_{1,2}$.

A saída então é arredondar este coeficiente para o inteiro mais próximo. Intuitivamente, ao fazer isso estamos projetando $\mathbf{b}_2$ no *espaço mais próximo* de $\text{span}(\mathbf{b}_1)^{\perp}$, de modo que se obtenha uma combinação linear de coeficientes inteiros em $\mathbf{b}_1$ e $\mathbf{b}_2$, ou seja, um ponto do reticulado.

Há mais um detalhe nesta ideia de "quase-ortogonalização". Se aplicarmos este procedimento de *redução de tamanho* apenas uma vez, podemos perder a chance de reduzir mais a base se o vetor $\mathbf{b}_1$ passar a ser maior do que $\mathbf{b}_2$. Então, depois

de cada redução, verificamos se $||\mathbf{b}_1|| > ||\mathbf{b}_2||$. Em caso afirmativo, trocamos $\mathbf{b}_1$ e $\mathbf{b}_2$ de posição e executamos novamente a redução. O procedimento para quando, mesmo após a redução, tivermos $||\mathbf{b}_2|| > ||\mathbf{b}_1||$.

Este procedimento, conhecido como *algoritmo de Gauss*, resolve de forma exata o SVP, uma vez que após o seu término, $\mathbf{b}_1$ é um vetor mais curto no reticulado gerado por B.

Algoritmo 2: Redução Gaussiana

Entrada: Uma base B de um reticulado em duas dimensões.
Saída: Uma base reduzida para o reticulado.
início
  $r \leftarrow 1$;
  enquanto $r \neq 0$ **faça**
    se $||\mathbf{b}_2|| < ||\mathbf{b}_1||$ **então**
      $\mathbf{b}_1 \leftrightarrow \mathbf{b}_2$;
    fim se
    $r \leftarrow \lfloor \mu_{1,2} \rceil$;
    $\mathbf{b}_2 \leftarrow \mathbf{b}_2 - r\mathbf{b}_1$;
  fim enqto
fim

Observe que o critério de parada do algoritmo é quando $r = 0$, ou seja, quando $|\mu_{1,2}| < 1/2$. Isto significa que $\mathbf{b}_2$ não pode mais ser reduzido por combinações lineares inteiras com $\mathbf{b}_1$.

3.3.2 Bases de Gram-Schmidt e o Volume de um Reticulado

Conforme já foi discutido no Capítulo 2, a partir de qualquer base B, é possível obter uma base ortogonal B^* tal que $\text{span}(B) = \text{span}(B^*)$. No entanto, de modo geral B^* não é uma base para $L(B)$, porque seus vetores são combinações lineares com coeficientes não necessariamente inteiros dos vetores de B. Ainda assim, é possível relacionar o determinante de $L(B)$ com os vetores da base de Gram-Schmidt.

Proposição 3.3 *Seja B uma base, e B^* a base de Gram-Schmidt associada. Então,*

$$\det(L(B)) = \prod_{i=1}^{n} ||\mathbf{b}_i^*||.$$

Demonstração: Da proposição 3.1, segue que $\det(L(B)) = |\det(B)|$. Mas $B = B^*M$, em que $\det(M) = 1$. Logo

$$\det(L(B)) = |\det(B)| = |\det(M)\det(B^*)| = |\det(B^*)| = \prod_{i=1}^{n} ||\mathbf{b}_i^*||. \square$$

A proposição que acabamos de demonstrar nos diz que, embora a base de Gram-Schmidt não seja necessariamente uma base para o reticulado, ainda assim o volume do paralelepípedo formado por seus vetores é sempre igual ao daquele formado pelos vetores de qualquer base do reticulado.

3.3.3 Bases Lovász-Reduzidas

Definição 3.2 *Uma base B de um reticulado é dita Lovász-reduzida se satisfaz as seguintes condições:*

$$|\mu_{i,j}| \leq \frac{1}{2}, \text{ para todos } 1 \leq i < j \leq n, \tag{3.3}$$

em que $\mu_{i,j}$ são os coeficientes de Gram-Schmidt, e

$$||\mathbf{b}_j^* + \mu_{j-1,j}\mathbf{b}_{j-1}^*||^2 \geq \delta||\mathbf{b}_{j-1}^*||^2, \text{ para todo } 1 < j \leq n, \tag{3.4}$$

em que $\mathbf{b}_j^*$ são os vetores da base ortogonal obtida pelo processo de Gram-Schmidt e $1/4 < \delta \leq 1$.

A expressão (3.3) é a chamada *condição de tamanho*. Uma base que satisfaça (3.3) é dita *reduzida de tamanho*. Por outro lado, (3.4) é conhecida como *condição*

de Lovász. Observe que, como a base de Gram-Schmidt é ortogonal, a condição (3.4) pode ser reescrita da seguinte maneira:

$$||\mathbf{b}_j^*||^2 \geq \left(\delta - \mu_{j-1,j}^2\right) ||\mathbf{b}_{j-1}^*||^2, \text{ para todo } 1 < j \leq n. \tag{3.5}$$

Por (3.3), temos que $\delta - \mu_{j-1,j}^2 \geq \delta - \frac{1}{4}$. Será conveniente, daqui para frente, fixar

$$\alpha = \frac{1}{\delta - \frac{1}{4}}.$$

Observe que, por definição, $\alpha \geq 4/3$. A fim de obter uma cota polinomial para o tempo de execução do LLL, utiliza-se $\delta = 3/4$. A seguir demonstramos que as bases Lovász-reduzidas possuem certas propriedades desejáveis. Nosso objetivo é mostrar que o primeiro vetor de uma base Lovász-reduzida é uma aproximação do vetor mais curto no reticulado.

Lema 3.3.1 *Seja B uma base Lovász-reduzida, e B^* a base de Gram-Schmidt associada. Então,*

$$||\mathbf{b}_i^*||^2 \leq \alpha^{j-i}||\mathbf{b}_j^*||^2,$$

para todo $i \leq j$.

Demonstração: faremos a demonstração por indução em i. Por (3.3) e (3.5), temos

$$||\mathbf{b}_{j-1}^*||^2 \leq \alpha||\mathbf{b}_j^*||^2.$$

Isto nos dá a base da indução ($i = j - 1$). Agora suponha que $||\mathbf{b}_i^*||^2 \leq \alpha^{j-i}||\mathbf{b}_j^*||^2$, para algum $i < j$. Usando a hipótese de indução:

$$||\mathbf{b}_{i-1}^*||^2 \leq \alpha||\mathbf{b}_i^*||^2 \leq \alpha\alpha^{(j-i)}||\mathbf{b}_j^*||^2 \leq \alpha^{j-(i-1)}||\mathbf{b}_j^*||^2.$$

□

Lema 3.3.2 *Se B é uma base Lovász-reduzida, então*

$$\prod_{j=1}^{n} ||\mathbf{b}_j||^2 \leq \alpha^{n(n-1)/2} \det^2(L(B)).$$

Demonstração: pela definição dos vetores de Gram-Schmidt, temos que

$$||\mathbf{b}_j||^2 = \left\| \mathbf{b}_j^* + \sum_{i=1}^{j-1} \mu_{i,j} \mathbf{b}_i^* \right\|^2 \leq ||\mathbf{b}_j^*||^2 + \sum_{i=1}^{j-1} \mu_{i,j}^2 ||\mathbf{b}_i^*||^2 \leq ||\mathbf{b}_j^*||^2 + \sum_{i=1}^{j-1} \frac{1}{4} ||\mathbf{b}_i^*||^2.$$

Mas, pelo Lema 3.3.1,

$$||\mathbf{b}_j||^2 \leq ||\mathbf{b}_j^*||^2 + \sum_{i=1}^{j-1} \frac{1}{4} \alpha^{j-i} ||\mathbf{b}_j^*||^2 = ||\mathbf{b}_j^*||^2 + ||\mathbf{b}_j^*||^2 \frac{\alpha^j}{4} \sum_{i=1}^{j-1} \alpha^{-i}.$$

Logo,

$$||\mathbf{b}_j||^2 \leq ||\mathbf{b}_j^*||^2 \left(1 + \frac{\alpha^{j-1}(1 - \alpha^{1-j})}{4(1 - \alpha^{-1})} \right) \leq \alpha^{j-1} ||\mathbf{b}_j^*||^2.$$

Como $\alpha \geq 4/3$, temos que $4(1 - \alpha^{-1}) \geq 1$. Portanto:

$$||\mathbf{b}_j||^2 \leq \alpha^{j-1} ||\mathbf{b}_j^*||^2. \tag{3.6}$$

Fazendo o produtório em (3.6) de 1 até n, obtemos

$$\prod_{j=1}^{n} ||\mathbf{b}_j||^2 \leq \prod_{j=1}^{n} \alpha^{j-1} ||\mathbf{b}_j^*||^2 = \alpha^{n(n-1)/2} \prod_{j=1}^{n} ||\mathbf{b}_j^*||^2.$$

Pela proposição 3.3, temos finalmente que

$$\prod_{j=1}^{n} ||\mathbf{b}_j||^2 \leq \alpha^{n(n-1)/2} \det^2(L(B)).$$

□

Lema 3.3.3 *Seja B uma base Lovász-reduzida, e B^* a base de Gram-Schmidt associada. Então,*

$$||\mathbf{b}_i||^2 \leq \alpha^{j-1} ||\mathbf{b}_j^*||^2,$$

para todo $i \leq j$.

Demonstração: da expressão (3.6) do lema anterior e do Lema 3.3.1, temos, para todo $i \leq j$,

$$||\mathbf{b}_i||^2 \leq \alpha^{i-1}||\mathbf{b}_i^*||^2 \leq \alpha^{i-1}\alpha^{j-i}||\mathbf{b}_j^*||^2.$$

Logo,

$$||\mathbf{b}_i||^2 \leq \alpha^{j-1}||\mathbf{b}_j^*||^2.$$

□

Chegamos agora ao resultado principal desta subseção. O teorema a seguir afirma que, em uma base Lovász-reduzida, o primeiro vetor corresponde a uma aproximação para um vetor mais curto no reticulado. Dito de outra forma, a redução de base resolve, de maneira aproximada, o SVP.

Teorema 3.3.1 *Seja B uma base Lovász-reduzida, e denote por λ o comprimento de um vetor mais curto (diferente de $\mathbf{0}$) em $L(B)$. Então*

$$||\mathbf{b}_1||^2 \leq \alpha^{n-1}\lambda^2.$$

Demonstração: Seja $\mathbf{v}$ um vetor mais curto em $L(B)$. Escrevendo $\mathbf{v}$ nas bases B e B^*, temos

$$\mathbf{v} = \sum_{j=1}^{n} a_j\mathbf{b}_j = \sum_{j=1}^{n} c_j\mathbf{b}_j^*,$$

em que $a_j \in \mathbb{Z}$ e $c_j \in \mathbb{R}$. Seja i tal que $a_j = 0$ para todo $j > i$. Então:

$$\sum_{j=1}^{i} a_j \sum_{k=1}^{j} \mu_{k,j}\mathbf{b}_k^* = \sum_{j=1}^{n} c_j\mathbf{b}_j^*.$$

No lado esquerdo da igualdade acima, o único coeficiente de $\mathbf{b}_i^*$ é a_i, de modo que $a_i = c_i$. Portanto:

$$||\mathbf{v}||^2 = \sum_{j=1}^{n} c_j^2||\mathbf{b}_j^*||^2 \geq c_i^2||\mathbf{b}_i^*||^2 \geq ||\mathbf{b}_i^*||^2,$$

pois $|a_i| = |c_i| \geq 1$. Do Lema 3.3.3, temos

$$||\mathbf{b}_i^*||^2 \geq \alpha^{1-i}||\mathbf{b}_1||^2.$$

Logo:

$$\lambda^2 = ||\mathbf{v}||^2 \geq \alpha^{1-i}||\mathbf{b}_1||^2 \geq \alpha^{1-n}||\mathbf{b}_1||^2 \Rightarrow ||\mathbf{b}_1||^2 \leq \alpha^{n-1}\lambda^2.$$

□

Para $\delta = 3/4$, temos que $||\mathbf{b}_1||^2 \leq 2^{n-1}\lambda^2(L(B))$. Esta cota pode ser melhorada no caso limite, em que $\delta = 1$:

$$||\mathbf{b}_1||^2 \leq \left(\frac{4}{3}\right)^{n-1} \lambda^2(L(B)).$$

No entanto, não se sabe se o LLL (próxima subseção) termina em tempo polinomial rodando com $\delta = 1$. Na prática, o que se faz é fixar um valor muito próximo de 1, por exemplo $\delta = 0.99$.

3.3.4 O Algoritmo LLL

O LLL permite obter uma base Lovász-reduzida para um reticulado de dimensão qualquer, empregando a ideia central do método de redução gaussiana, com alguns detalhes adicionais conforme é mostrado no Algoritmo 3.

3.3.4.1 A função incrementalGramSchmidt

A definição de base Lovász-reduzida depende da base de Gram-Schmidt associada. Portanto, uma das etapas essenciais do algoritmo LLL consiste em executar a ortogonalização, a fim de que possam ser testadas as condições de tamanho e de

Algoritmo 3: LLL(B)

```
Entrada: Uma base B de um reticulado.
Saída: A base B Lovász-reduzida.
início
    j ← 2;
    j_max ← 1;
    enquanto j ≤ n faça
        se j > j_max então
            incrementalGramSchmidt(B, j);
            j_max ← j;
        fim se
        i ← j − 1;
        reduzir(B, i, j);
        se testeLovasz(B, j) = Falso então
            swapLovasz(B, j);
            se j > 2 então
                j ← j − 1;
            fim se
        senão
            para i = j − 2, ··· , 1 faça
                reduzir(B, i, j);
            fim para
            j ← j + 1;
        fim se
    fim enqto
fim
```

Lovász. No entanto, a condição (3.5) pode ser testada apenas com as normas quadradas dos vetores de Gram-Schmidt, sendo portanto desnecessário armazenar os vetores em si.

Assim é preciso calcular os coeficientes e as normas quadradas dos vetores de Gram-Schmidt, sem que estes precisem ser armazenados. No modo *tradicional* de execução do algoritmo, isto não é possível, porque cada coeficiente depende explicitamente de um vetor da base ortogonal:

$$\mu_{i,j} = \frac{\langle \mathbf{b}_i^*, \mathbf{b}_j \rangle}{||\mathbf{b}_i^*||^2}.$$

No entanto, se fizermos $B_i = ||\mathbf{b}_i^*||^2$ e utilizarmos a definição dos vetores de Gram-Schmidt, obtemos

$$\mu_{i,j} = \frac{1}{B_i}\langle \mathbf{b}_j, \mathbf{b}_i - \sum_{k=1}^{i-1} \mu_{k,i}\mathbf{b}_k^* \rangle = \frac{1}{B_i}\left(\langle \mathbf{b}_i, \mathbf{b}_j \rangle - \sum_{k=1}^{i-1} \mu_{k,i}\langle \mathbf{b}_k^*, \mathbf{b}_j \rangle \right).$$

Mas observe que $\langle \mathbf{b}_k^*, \mathbf{b}_j \rangle = \mu_{k,j} B_k$, logo

$$\mu_{i,j} = \frac{1}{B_i} \left(\langle \mathbf{b}_i, \mathbf{b}_j \rangle - \sum_{k=1}^{i-1} \mu_{k,i} \mu_{k,j} B_k \right). \tag{3.7}$$

O próximo passo é calcular os quadrados das normas dos vetores de Gram-Schmidt:

$$B_k = \langle \mathbf{b}_k^*, \mathbf{b}_k^* \rangle = \langle \mathbf{b}_k - \sum_{j=1}^{k-1} \mu_{j,k} \mathbf{b}_j^*, \mathbf{b}_k - \sum_{j=1}^{k-1} \mu_{j,k} \mathbf{b}_j^* \rangle.$$

Como os vetores de Gram-Schmidt são dois a dois ortogonais, obtemos

$$B_k = \langle \mathbf{b}_k, \mathbf{b}_k \rangle - 2 \sum_{j=1}^{k-1} \mu_{j,k} \langle \mathbf{b}_k, \mathbf{b}_j^* \rangle + \sum_{j=1}^{k-1} \mu_{j,k}^2 B_j,$$

donde obtemos, finalmente,

$$B_j = \langle \mathbf{b}_j, \mathbf{b}_j \rangle - \sum_{k=1}^{j-1} \mu_{k,j}^2 B_k. \tag{3.8}$$

As expressões (3.7) e (3.8) permitem definir a função incrementalGramSchmidt para o cálculo dos coeficientes da j-ésima coluna e de B_j.

Algoritmo 4: incrementalGramSchmidt(B, j)

Entrada: Uma base B e um inteiro j
Saída: Os coeficientes de Gram-Schmidt da j-ésima coluna e B_j.
início
  para $i = 1, \cdots, j$ **faça**
    $\lambda \leftarrow \langle \mathbf{b}_i, \mathbf{b}_j \rangle$;
    para $k = 1, \cdots, i-1$ **faça**
      $\lambda \leftarrow \lambda - \mu_{k,i} \mu_{k,j} B_k$;
    fim para
    se $i < j$ **então**
      $\mu_{i,j} \leftarrow \lambda / B_i$;
    senão
      $B_j \leftarrow \lambda$;
    fim se
  fim para
fim

Observe que, no Algoritmo 4, o cálculo de $\mu_{i,j}$ e B_j depende dos coeficientes das linhas anteriores à i-ésima linha, nas colunas i e j, e dos valores de B_k das colunas anteriores. Portanto, na execução do LLL, estes dados podem ser calculados *sob demanda*, evitando o armazenamento precoce de dados que não serão usados na

iteração atual. Em outras palavras, na j-ésima iteração do LLL, o Algoritmo 4 calcula somente os dados de Gram-Schmidt da j-ésima coluna. Esta é a razão pela qual [40] se refere a este algoritmo como *incremental*.

3.3.4.2 A função reduzir

A primeira condição que caracteriza uma base reduzida, conhecida como *condição de tamanho*, envolve os coeficientes de Gram-Schmidt e pode ser interpretada de maneira simples. Primeiro, observe que

$$|\mu_{i,j}| \leq \frac{1}{2} \Rightarrow ||\mathbf{b}_j|||\cos\theta| \leq \frac{||\mathbf{b}_i^*||}{2},$$

em que θ é o ângulo entre $\mathbf{b}_j$ e $\mathbf{b}_i^*$. Por outro lado, seja γ o ângulo entre $\mathbf{b}_i$ e $\mathbf{b}_i^*$. Então

$$\langle \mathbf{b}_i, \mathbf{b}_i^* \rangle = ||\mathbf{b}_i||||\mathbf{b}_i^*||\cos\gamma \Rightarrow \cos\gamma = \frac{\langle \mathbf{b}_i, \mathbf{b}_i^* \rangle}{||\mathbf{b}_i||||\mathbf{b}_i^*||} = \frac{||\mathbf{b}_i^*||}{||\mathbf{b}_i||}.$$

Portanto, $||\mathbf{b}_i^*|| = ||\mathbf{b}_i||\cos\gamma$. Substituindo na desigualdade anterior, obtemos

$$||\mathbf{b}_j|||\cos\theta| \leq \frac{||\mathbf{b}_i|||\cos\gamma|}{2}.$$

Assim, a condição de tamanho nos diz que a projeção de $\mathbf{b}_j$ sobre $\mathbf{b}_i^*$ não é maior do que metade da projeção de $\mathbf{b}_i$ sobre $\mathbf{b}_i^*$. Então, ou $\mathbf{b}_j$ é suficientemente *pequeno* em relação a $\mathbf{b}_i$, ou então o ângulo entre esses vetores é suficientemente *grande*. Esta é uma interpretação razoável diante daquilo que esperamos de uma base *boa*.

Toda vez que a condição de tamanho falha para o coeficiente $\mu_{i,j}$, o vetor $\mathbf{b}_j$ é *reduzido* em relação a $\mathbf{b}_i$, na etapa conhecida como *redução de tamanho*:

$$\mathbf{b}'_j = \mathbf{b}_j - r\mathbf{b}_i, \tag{3.9}$$

em que $\mathbf{b}'_j$ é o novo vetor $\mathbf{b}_j$, e r é o inteiro mais próximo de $\mu_{i,j}$.

Após esta mudança, os coeficientes de Gram-Schmidt na coluna j precisam ser atualizados. De fato, observe que o novo valor de $\mu_{i,j}$ é dado por

$$\mu'_{i,j} = \frac{\langle \mathbf{b}_i^*, \mathbf{b}_j \rangle}{B_i} - r\frac{\langle \mathbf{b}_i^*, \mathbf{b}_i \rangle}{B_i}.$$

Como $\mathbf{b}_i$ é combinação linear de $\mathbf{b}_i^*$ e dos vetores $\mathbf{b}_k^*$, para $k = 1, \cdots, i-1$, temos que $\langle \mathbf{b}_i^*, \mathbf{b}_i \rangle = \langle \mathbf{b}_i^*, \mathbf{b}_i^* \rangle = B_i$. Logo, na redução de tamanho, o coeficiente $\mu_{i,j}$ é atualizado da seguinte maneira:

$$\mu'_{i,j} = \mu_{i,j} - r. \tag{3.10}$$

Por outro lado, para $k = 1, \cdots, i-1$, temos

$$\mu'_{k,j} = \frac{\langle \mathbf{b}_k^*, \mathbf{b}_j \rangle}{B_k} - r\frac{\langle \mathbf{b}_k^*, \mathbf{b}_i \rangle}{B_k}.$$

Portanto, para $k = 1, \cdots, i-1$, é feita a seguinte atualização:

$$\mu'_{k,j} = \mu_{k,j} - r\mu_{k,i}. \tag{3.11}$$

É importante observar que, após as atualizações acima, apenas o coeficiente $\mu_{i,j}$ passa a satisfazer necessariamente a condição de tamanho. Com as equações (3.9), (3.10) e (3.11), podemos definir a função reduzir:

Algoritmo 5: reduzir(B, i, j)

Entrada: Uma base B e dois inteiros i e j
Saída: Os coeficientes da coluna j atualizados.
início
  $r \leftarrow \lceil \mu_{i,j} \rfloor$;
  $\mathbf{b}_j \leftarrow \mathbf{b}_j - r\mathbf{b}_i$;
  $\mu_{i,j} \leftarrow \mu_{i,j} - r$;
  para $k = 1, \cdots, i-1$ **faça**
    $\mu_{k,j} \leftarrow \mu_{k,j} - r\mu_{k,i}$;
  fim para
fim

3.3.4.3 A função swapLovasz

A condição de Lovász implica em que os vetores da base de Gram-Schmidt obedecem a uma certa limitação de tamanho relativo, de modo que a componente

de $\mathbf{b}_j$ ortogonal a span$(\mathbf{b}_1, \cdots, \mathbf{b}_{j-2})$ não pode ser muito menor do que $\mathbf{b}^*_{j-1}$, isto é, a componente de $\mathbf{b}_{j-1}$ ortogonal a span$(\mathbf{b}_1, \cdots, \mathbf{b}_{j-2})$. Então podemos interpretar a condição de Lovász como uma imposição sobre a ordem nos vetores da base do reticulado. Assim, toda vez que ela não for satisfeita, os vetores $\mathbf{b}_j$ e $\mathbf{b}_{j-1}$ serão trocados de posição.

A função testeLovasz simplesmente testa a condição (3.5), retornando True se ela for satisfeita, e False caso contrário.

Algoritmo 6: testeLovasz(j)

```
Entrada: Um inteiro j.
Saída: Verdadeiro ou Falso
início
    se B_j ≥ (δ − μ²_{j−1,j})B_{j−1} então
        retorna True
    senão
        retorna False
    fim se
fim
```

Se o teste acima retornar False, os vetores $\mathbf{b}_j$ e $\mathbf{b}_{j-1}$ são trocados de posição. Obviamente, isto exige que se atualizem os coeficientes de Gram-Schmidt e os valores de B_j e B_{j-1}.

Inicialmente é preciso atualizar os coeficientes $\mu_{i,j-1}$, para $i = 1, \cdots, j-2$:

$$\mu'_{i,j-1} = \frac{1}{B_i}\left(\langle \mathbf{b}_i, \mathbf{b}_j\rangle - \sum_{k=1}^{i-1} \mu_{k,i}\mu_{k,j}B_k\right),$$

ou seja,

$$\mu'_{i,j-1} = \mu_{i,j}. \tag{3.12}$$

O valor atualizado de B_{j-1} será então dado por

$$B'_{j-1} = \langle \mathbf{b}_j, \mathbf{b}_j\rangle - \sum_{k=1}^{j-2} \mu^2_{k,j}B_k,$$

que é equivalente a

$$B'_{j-1} = B_j + \mu^2_{j-1,j}B_{j-1}. \tag{3.13}$$

O próximo passo é atualizar os coeficientes $\mu_{i,j}$, inicialmente para $i = 1, \cdots, j-2$:

$$\mu'_{i,j} = \frac{1}{B_i}\left(\langle \mathbf{b}_i, \mathbf{b}_{j-1}\rangle - \sum_{k=1}^{i-1} \mu_{k,i}\mu_{k,j-1}B_k\right),$$

ou seja,

$$\mu'_{i,j} = \mu_{i,j-1}. \tag{3.14}$$

A atualização de $\mu_{j-1,j}$ depende do novo valor de B_{j-1}:

$$\mu'_{j-1,j} = \frac{1}{B'_{j-1}}\left(\langle \mathbf{b}_{j-1}, \mathbf{b}_j\rangle - \sum_{k=1}^{j-2} \mu_{k,j}\mu_{k,j-1}B_k\right).$$

Portanto,

$$\mu'_{j-1,j} = \mu_{j-1,j}\frac{B_{j-1}}{B'_{j-1}}. \tag{3.15}$$

Finalmente podemos calcular o valor de B'_j:

$$B'_j = \langle \mathbf{b}_{j-1}, \mathbf{b}_{j-1}\rangle - \sum_{i=1}^{j-2} \mu^2_{i,j-1}B_i - \mu^2_{j-1,j}\frac{B^2_{j-1}}{B'_{j-1}} = B_{j-1} - \mu^2_{j-1,j}\frac{B^2_{j-1}}{B'_{j-1}}.$$

Logo:

$$B'_j = \frac{B_{j-1}B_j}{B'_{j-1}}. \tag{3.16}$$

Ainda é preciso cuidar dos coeficientes $\mu_{j-1,k}$ e $\mu_{j,k}$, para $k > j$, pois eles dependem dos valores atualizados B'_{j-1} e B'_j. De fato,

$$\mu'_{j-1,k} = \frac{1}{B'_{j-1}}\left(\langle \mathbf{b}_j, \mathbf{b}_k\rangle - \sum_{i=1}^{j-2} \mu_{i,k}\mu_{i,j}B_i\right) = \frac{\mu_{j,k}B_j + \mu_{j-1,j}\mu_{j-1,k}B_{j-1}}{B'_{j-1}}.$$

Utilizando as expressões (3.13) e (3.15), obtemos

$$\mu'_{j-1,k} = \frac{\mu_{j,k}B'_{j-1} - \mu_{j,k}\mu^2_{j-1,j}B_{j-1}}{B'_{j-1}} + \mu_{j-1,k}\mu'_{j-1,j}.$$

Portanto,

$$\mu'_{j-1,k} = \mu_{j,k} - \mu_{j,k}\mu_{j-1,j}\mu'_{j-1,j} + \mu_{j-1,k}\mu'_{j-1,j}. \tag{3.17}$$

Em seguida, atualizamos os coeficientes $\mu_{j,k}$, para $k > j$:

$$\mu'_{j,k} = \frac{1}{B'_j}\left(\langle \mathbf{b}_{j-1}, \mathbf{b}_k\rangle - \sum_{i=1}^{j-2} \mu_{i,j-1}\mu_{i,k}B_i - \mu'_{j-1,j}\mu'_{j-1,k}B'_{j-1}\right).$$

Após algumas transformações, a expressão acima pode ser reescrita como

$$\mu'_{j,k} = (\mu_{j-1,k} - \mu_{j,k}\mu_{j-1,j}) \left(\frac{B_{j-1}}{B'_j} - (\mu'_{j-1,j})^2 \frac{B'_{j-1}}{B'_j} \right),$$

e usando o fato de que $B'_j = B_{j-1} - (\mu'_{j-1,j})^2 B'_{j-1}$, obtemos finalmente

$$\mu'_{j,k} = \mu_{j-1,k} - \mu_{j,k}\mu_{j-1,j}. \tag{3.18}$$

As expressões de (3.12) a (3.18) correspondem às atualizações que devem ser feitas na etapa do *swap* de Lovász, e estão resumidas no algoritmo abaixo:

Algoritmo 7: swapLovasz(j)

Entrada: Um inteiro j.
Saída: Os dados de Gram-Schmidt atualizados.
início
  $\mathbf{b}_{j-1} \leftrightarrow \mathbf{b}_j$;
  para $i = 1, \cdots, j-2$ **faça**
    $\mu_{i,j-1} \leftrightarrow \mu_{i,j}$;
  fim para
  $b \leftarrow B_{j-1}$;
  $B_{j-1} \leftarrow B_j + \mu_{j-1,j}^2 B_{j-1}$;
  $\lambda_1 \leftarrow \mu_{j-1,j}$;
  $\mu_{j-1,j} \leftarrow b\mu_{j-1,j}/B_{j-1}$;
  $B_j \leftarrow bB_j/B_{j-1}$;
  para $k = j+1, \cdots, n$ **faça**
    $\lambda_2 \leftarrow \mu_{j-1,k}$;
    $\mu_{j-1,k} \leftarrow \mu_{j,k} - \lambda_1\mu_{j,k}\mu_{j-1,j} + \mu_{j-1,k}\mu_{j-1,j}$;
    $\mu_{j,k} \leftarrow \lambda_2 - \lambda_1\mu_{j,k}$;
  fim para
fim

3.3.5 O LLL Inteiro

O LLL pode apresentar problemas de precisão nos cálculos dos coeficientes e da base de Gram-Schmidt. Pode parecer tentador implementá-lo com aritmética de números racionais, mas isto compromete consideravelmente o desempenho do algoritmo, uma vez que os numeradores e denominadores se tornam excessivamente grandes à medida que o algoritmo é executado. Ainda assim, existem maneiras de minimizar os problemas de precisão. A primeira consiste em substituir a fórmula (3.7) pela expressão

$$\gamma_{i,j} = \langle \mathbf{b}_i, \mathbf{b}_j \rangle - \sum_{k=1}^{i-1} \mu_{k,i}\gamma_{k,j}, \tag{3.19}$$

e fazer $\mu_{i,j} = \gamma_{i,j}/B_i$. Seguindo a mesma linha de raciocínio, a fórmula (3.8) pode ser substituída por

$$B_j = \langle \mathbf{b}_j, \mathbf{b}_j \rangle - \sum_{k=1}^{j-1} \mu_{k,j}\gamma_{k,j}. \tag{3.20}$$

As expressões acima reduzem o número de operações realizadas durante a etapa de Gram-Schmidt, diminuindo consequentemente os problemas de precisão.

A segunda maneira de contornar os problemas de precisão é realizar todos os cálculos do algoritmo com números inteiros, mas isto somente é possível quando a base do reticulado é inteira. Os procedimentos apresentados a seguir são essencialmente os mesmos descritos por [40].

Para tirarmos vantagem do fato de que a base do reticulado é inteira, precisamos recorrer à Proposição 3.3, que estabelece uma relação entre o determinante de um reticulado $L(B)$ e os vetores da base de Gram-Schmidt associada. Lembremos que

$$\det^2(L(B)) = \det((\langle \mathbf{b}_i, \mathbf{b}_j \rangle)_{1 \leq i,j \leq n}) = \prod_{j=1}^{n} B_j,$$

em que $B_j = ||\mathbf{b}_j^*||^2$. Observe que a expressão acima é inteira, já que a base do reticulado é inteira. Então, podemos fixar $D_0 = 1$ e definir

$$D_k = \det((\langle \mathbf{b}_i, \mathbf{b}_j \rangle)_{1 \leq i,j \leq k}) = \prod_{j=1}^{k} B_j. \tag{3.21}$$

Claramente temos que $D_k \in \mathbb{Z}$, para $k = 1, \cdots, n$. Observe ainda que $D_k = D_{k-1}B_k$, de modo que D_k e D_{k-1} correspondem respectivamente ao numerador e ao denominador (ambos inteiros) de B_k. Então, em vez de calcular diretamente os quadrados das normas dos vetores de Gram-Schmidt, podemos armazenar os valores D_k, para $k = 1, \cdots, n$.

A proposição a seguir estabelece valores inteiros que serão utlizados para o cálculo dos coeficientes de Gram-Schmidt.

Proposição 3.4 *Seja $i < j$, e D_i conforme definido na expressão* (3.21)*. Então $D_i\mu_{i,j} \in \mathbb{Z}$. Além disso,*

$$D_i \sum_{k=1}^{i} \mu_{k,j}\mu_{k,l}B_k \in \mathbb{Z},$$

para $l = i+1, \cdots, j$.

Demonstração: defina o vetor

$$\mathbf{v} = \mathbf{b}_j - \sum_{k=1}^{i} \mu_{k,j}\mathbf{b}_k^* = \mathbf{b}_j^* + \sum_{k=i+1}^{j-1} \mu_{k,j}\mathbf{b}_k^*.$$

Observe que $\langle \mathbf{b}_l^*, \mathbf{v}\rangle = 0$, para $l = 1, \cdots, i$. Então

$$\langle \mathbf{b}_l, \mathbf{v}\rangle = 0, \tag{3.22}$$

para $l = 1, \cdots, i$. Por outro lado, podemos escrever

$$\mathbf{v} = \mathbf{b}_j - \sum_{k=1}^{i} \sigma_k\mathbf{b}_k.$$

Substituindo na expressão (3.22), obtemos

$$\sum_{k=1}^{i} \sigma_k\langle \mathbf{b}_l, \mathbf{b}_k\rangle = \langle \mathbf{b}_l, \mathbf{b}_j\rangle, \text{ para } l = 1, \cdots, i. \tag{3.23}$$

Escrevendo (3.23) na forma matricial, temos

$$\begin{pmatrix} \langle \mathbf{b}_1, \mathbf{b}_1\rangle & \cdots & \langle \mathbf{b}_1, \mathbf{b}_i\rangle \\ \vdots & \ddots & \vdots \\ \langle \mathbf{b}_i, \mathbf{b}_1\rangle & \cdots & \langle \mathbf{b}_i, \mathbf{b}_i\rangle \end{pmatrix} \begin{pmatrix} \sigma_1 \\ \vdots \\ \sigma_i \end{pmatrix} = \begin{pmatrix} \langle \mathbf{b}_1, \mathbf{b}_j\rangle \\ \vdots \\ \langle \mathbf{b}_i, \mathbf{b}_j\rangle \end{pmatrix}.$$

A solução do sistema acima, para $k = 1, \cdots, i$, é dada por $\sigma_k = \delta_k/D_i$, em que δ_k é um inteiro. Então, em particular, $D_i\sigma_i = \delta_i \in \mathbb{Z}$. Mas observe que

$$\sum_{k=1}^{i} \sigma_k\mathbf{b}_k = \sum_{k=1}^{i} \mu_{k,j}\mathbf{b}_k^*.$$

Fazendo o produto interno por $\mathbf{b}_i^*$ em ambos os lados, obtemos

$$\sigma_i\langle \mathbf{b}_i, \mathbf{b}_i^*\rangle = \mu_{i,j}\langle \mathbf{b}_i^*, \mathbf{b}_i^*\rangle.$$

Logo, $\sigma_i = \mu_{i,j}$. Então, $D_i\sigma_i = D_i\mu_{i,j} \in \mathbb{Z}$.

Observe que acabamos de mostrar que $D_i\mathbf{v}$ é um vetor do reticulado, de modo que $\langle D_i\mathbf{v}, \mathbf{b}_l\rangle \in \mathbb{Z}$, para $l = 1, \cdots, n$. Portanto,

$$D_i\langle \mathbf{b}_j, \mathbf{b}_l\rangle - D_i \sum_{k=1}^{i} \mu_{k,j}\langle \mathbf{b}_k^*, \mathbf{b}_l\rangle \in \mathbb{Z}.$$

Em particular, para todo $i < l \leq j$, temos que

$$D_i \sum_{k=1}^{i} \mu_{k,j}\langle \mathbf{b}_k^*, \mathbf{b}_l\rangle = D_i \sum_{k=1}^{i} \mu_{k,j}\mu_{k,l}B_k \in \mathbb{Z}.$$

□

Podemos definir então o número inteiro $\lambda_{i,j} = D_i\mu_{i,j}$, fixando $\lambda_{j,j} = D_j$. Assim temos

$$\lambda_{i,j} = D_{i-1}\left(\langle \mathbf{b}_i, \mathbf{b}_j\rangle - \sum_{k=1}^{i-1} \frac{\lambda_{k,i}\lambda_{k,j}B_k}{D_k^2}\right) = D_{i-1}\left(\langle \mathbf{b}_i, \mathbf{b}_j\rangle - \sum_{k=1}^{i-1} \frac{\lambda_{k,i}\lambda_{k,j}}{D_kD_{k-1}}\right).$$

O valor de $\lambda_{i,j}$ pode então ser calculado iterativamente, se definirmos $Z_0 = \langle \mathbf{b}_i, \mathbf{b}_j\rangle$ e, para cada $l = 1, \cdots, i-1$,

$$Z_l = D_l\left(\langle \mathbf{b}_i, \mathbf{b}_j\rangle - \sum_{k=1}^{l} \frac{\lambda_{k,i}\lambda_{k,j}}{D_kD_{k-1}}\right).$$

Para calcular recursivamente os valores de Z_l, observe que

$$Z_l = D_l\left(\langle \mathbf{b}_i, \mathbf{b}_j\rangle - \sum_{k=1}^{l-1} \frac{\lambda_{k,i}\lambda_{k,j}}{D_kD_{k-1}} - \frac{\lambda_{l,i}\lambda_{l,j}}{D_lD_{l-1}}\right) = \frac{D_lZ_{l-1}}{D_{l-1}} - \frac{\lambda_{l,i}\lambda_{l,j}}{D_{l-1}}.$$

Logo:

$$Z_l = \frac{D_lZ_{l-1} - \lambda_{l,i}\lambda_{l,j}}{D_{l-1}}, \tag{3.24}$$

em que $Z_0 = \langle \mathbf{b}_i, \mathbf{b}_j\rangle$ e $Z_{i-1} = \lambda_{i,j}$. Observe que, pela proposição anterior, cada Z_l é um número inteiro.

Além disso, cada D_j pode ser calculado iterativamente de maneira similar, pois

$$D_{j-1}B_j = D_j = D_{j-1}\left(\langle \mathbf{b}_j, \mathbf{b}_j\rangle - \sum_{k=1}^{j-1} \mu_{k,j}^2 B_k\right).$$

Podemos definir então, para $l = 1, \cdots, j-1$,

$$Y_l = D_l\left(\langle \mathbf{b}_j, \mathbf{b}_j\rangle - \sum_{k=1}^{l} \frac{\lambda_{k,j}^2}{D_k D_{k-1}}\right).$$

Calculando recursivamente, como no caso anterior, obtemos

$$Y_l = D_l\left(\langle \mathbf{b}_j, \mathbf{b}_j\rangle - \sum_{k=1}^{l-1} \frac{\lambda_{k,j}^2}{D_k D_{k-1}} - \frac{\lambda_{l,j}^2}{D_l D_{l-1}}\right) = \frac{D_l Y_{l-1}}{D_{l-1}} - \frac{\lambda_{l,j}^2}{D_{l-1}},$$

ou seja,

$$Y_l = \frac{D_l Y_{l-1} - \lambda_{l,j}^2}{D_{l-1}}. \tag{3.25}$$

em que $Y_0 = \langle \mathbf{b}_j, \mathbf{b}_j\rangle$ e $Y_{j-1} = D_j$. Observe, no entanto, que a expressão (3.25) é um caso particular da fórmula (3.24), para $i = j$. Assim, os valores de $\lambda_{i,j}$ e D_j podem ser calculados em conjunto, sem a necessidade de utilizar expressões diferentes.

Resta definir como serão testadas as condições de tamanho e de Lovász, já que os coeficientes e a base de Gram-Schmidt estão sendo calculados indiretamente. Como definimos $\lambda_{i,j} = D_i\mu_{i,j}$, para testar a condição de tamanho, basta verificar se a desigualdade $2|\lambda_{i,j}| \leq |D_i|$ é satisfeita. Para testar a condição de Lovász, verifica-se a desigualdade

$$\frac{D_j}{D_{j-1}} \geq \left(\frac{3}{4} - \frac{\lambda_{j,j-1}^2}{D_{j-1}^2}\right)\frac{D_{j-1}}{D_{j-2}}.$$

Como estamos trabalhando somente com valores inteiros, é conveniente não depender do uso de frações. Então a desigualdade acima pode ser reescrita da seguinte maneira:

$$4(D_j D_{j-2} + \lambda_{j,j-1}^2) \geq 3D_{j-1}^2. \tag{3.26}$$

A seguir discutiremos, de maneira muito mais breve, uma outra caracterização de base reduzida, que deu origem ao algoritmo de redução conhecido como BKZ.

3.3.6 Bases Korkine-Zolotareff-Reduzidas e o algoritmo BKZ

Nesta seção será apresentada uma noção diferente de base reduzida. Antes de prosseguirmos, convém estabelecer algumas definições e notações básicas.

Seja B uma base. O componente de $\mathbf{b}_j$ ortogonal a $\text{span}(\mathbf{b}_1, \cdots, \mathbf{b}_{i-1})$ será denotado por $\mathbf{b}_j(i)$. Segue desta definição que $\mathbf{b}_j(j) = \mathbf{b}_j^*$. Além disso, usando o fato de que

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

podemos facilmente concluir que

$$\mathbf{b}_j(i) = \sum_{k=i}^{j} \mu_{k,j} \mathbf{b}_k^*. \tag{3.27}$$

Vamos considerar o reticulado gerado pelos componentes dos vetores $\mathbf{b}_i, \cdots, \mathbf{b}_n$ ortogonais a $\text{span}(\mathbf{b}_1, \cdots, \mathbf{b}_{i-1})$ e denotá-lo por $L_i(B)$, ou seja,

$$L_i(B) = L([\mathbf{b}_i(i) \cdots \mathbf{b}_n(i)]).$$

Note que $L_i(B)$ é a projeção de $L(B)$ no complemento ortogonal do espaço gerado por $\mathbf{b}_1, \cdots, \mathbf{b}_{i-1}$. Observe também que $L_1(B) = L(B)$. Vamos definir então uma base Korkine-Zolotareff reduzida:

Definição 3.3 *Uma base B é dita Korkine-Zolotareff reduzida (KZ-reduzida) se for reduzida de tamanho e se $\mathbf{b}_i^*$ é o vetor mais curto de $L_i(B)$, para todo $i = 1, \cdots, n$.*

Já vimos que, se B é uma base Lovász-reduzida, $\mathbf{b}_1$ é apenas uma aproximação para o vetor mais curto de $L(B)$. Pela Definição 3.3, a noção de base KZ-reduzida é mais forte, pois implica em que $\mathbf{b}_1$ seja de fato o vetor mais curto de $L(B)$.

Definição 3.4 *Uma base B é dita β-reduzida se ela é reduzida de tamanho e se $\mathbf{b}_j(j), \cdots, \mathbf{b}_{j+\beta-1}(j)$ é uma base KZ-reduzida, para todo $j = 1, \cdots, n-\beta+1$.*

A sequência $\mathbf{b}_j(j), \cdots, \mathbf{b}_{j+\beta-1}(j)$ é denominada um β-bloco. Para $\beta = 2$, esta definição implica em

$$||\mathbf{b}_j(j)||^2 = ||\mathbf{b}_j^*||^2 \leq ||\mathbf{b}_{j+1}(j)||^2 = ||\mathbf{b}_{j+1}^* + \mu_{j,j+1}\mathbf{b}_j^*||^2.$$

Observe que esta condição é um caso particular de (3.4), para $\delta = 1$.

Definição 3.5 *Uma base $B = [\mathbf{b}_1 \cdots \mathbf{b}_{m\beta}]$ é dita 2β-reduzida em blocos se ela é reduzida de tamanho e se os 2β-blocos $\mathbf{b}_{j\beta+1}(j\beta+1), \cdots, \mathbf{b}_{(j+2)\beta}(j\beta+1)$ são KZ-reduzidos, para todo $j = 0, \cdots, m-2$.*

Toda base 2β-reduzida é 2β-reduzida em blocos, e toda base 2-reduzida em blocos é Lovász-reduzida. A qualidade das bases β-reduzidas está diretamente relacionada às constantes

$$\alpha_\beta = \max \frac{||\mathbf{b}_1||^2}{||\mathbf{b}_\beta^*||^2}, \tag{3.28}$$

em que o máximo é tomado sobre todas as bases $[\mathbf{b}_1 \cdots \mathbf{b}_\beta]$ KZ-reduzidas. Observe que, para $\beta = 2$, temos $\alpha_2 = 4/3$, que corresponde ao caso limite das bases Lovász-reduzidas, quando $\delta = 1$.

Observe ainda que $\alpha_\beta \leq \alpha_{\beta+1}$ para todo β, já que toda base KZ-reduzida $[\mathbf{b}_2 \cdots \mathbf{b}_{\beta+1}]$ pode ser estendida para uma base $[\mathbf{b}_1 \cdots \mathbf{b}_{\beta+1}]$, também KZ-reduzida, em que $\mathbf{b}_1 \in \text{span}(\mathbf{b}_2, \cdots, \mathbf{b}_{\beta+1})^\perp$ e $||\mathbf{b}_1|| = ||\mathbf{b}_2||$.

A seguir mostramos que, assim como as bases Lovász-reduzidas, o vetor mais curto de um reticulado pode ser aproximado pelo primeiro vetor de uma base β-reduzida.

Proposição 3.5 *Se* $[\mathbf{b}_1 \cdots \mathbf{b}_n]$ *é uma base* β*-reduzida, então*

$$||\mathbf{b}_1||^2 \leq \alpha_\beta^{(n-1)/(\beta-1)} \lambda^2,$$

em que λ *é o comprimento do vetor mais curto em* $L(B)$*.*

Demonstração: Seja $\mathbf{v}$ um vetor mais curto em $L(B)$. Escrevendo $\mathbf{v}$ nas bases B e B^*, temos

$$\mathbf{v} = \sum_{j=1}^{n} a_j \mathbf{b}_j = \sum_{j=1}^{n} c_j \mathbf{b}_j^*,$$

em que $a_j \in \mathbb{Z}$ e $c_j \in \mathbb{R}$. Seja i tal que $a_j = 0$ para todo $j > i$. Então:

$$\sum_{j=1}^{i} a_j \sum_{k=1}^{j} \mu_{k,j} \mathbf{b}_k^* = \sum_{j=1}^{n} c_j \mathbf{b}_j^*.$$

No lado esquerdo da igualdade acima, o único coeficiente de $\mathbf{b}_i^*$ é a_i, de modo que $a_i = c_i$. Portanto:

$$||\mathbf{v}||^2 = \sum_{j=1}^{n} c_j^2 ||\mathbf{b}_j^*||^2 \geq c_i^2 ||\mathbf{b}_i^*||^2 \geq ||\mathbf{b}_i^*||^2,$$

pois $|a_i| = |c_i| \geq 1$. Por outro lado, B satisfaz

$$||\mathbf{b}_j^*||^2 \leq \alpha_\beta ||\mathbf{b}_{j+l}^*||^2,$$

para $l \leq \beta - 1, j + l \leq n$. Por indução obtemos

$$||\mathbf{b}_j^*||^2 \leq \alpha_\beta^m ||\mathbf{b}_{j+l}^*||^2,$$

para $l \leq m(\beta - 1), j + l \leq n$. Assim temos

$$||\mathbf{b}_1||^2 = ||\mathbf{b}_1^*||^2 \leq \alpha_\beta^{(n-1)/(\beta-1)} ||\mathbf{b}_i^*||^2 \leq \alpha_\beta^{(n-1)/(\beta-1)} \lambda^2.$$

□

Em [41] esta cota foi melhorada para $||\mathbf{b}_1||^2 \leq \gamma_\beta^{2(n-1)/(\beta-1)}\lambda^2$, em que γ_β é a *constante de Hermite* em dimensão β.

Uma hierarquia de algoritmos de redução entre o LLL e a KZ-redução foi proposta em [42], juntamente com uma versão melhorada do algoritmo de [43] para KZ-redução. O BKZ foi uma versão melhorada do algoritmo para KZ-redução em blocos.

O BKZ realiza uma β-redução com parâmetro $1/2 < \delta < 1$. Uma base é dita β-reduzida com parâmetro δ se

$$\delta||\mathbf{b}_i^*||^2 \leq \lambda(L_i(\mathbf{b}_1, \cdots, \mathbf{b}_{\min(i+\beta-1,n)}))^2,$$

para $i = 1, \cdots, n-1$. Nesta definição, $\lambda(L_i(\mathbf{b}_1, \cdots, \mathbf{b}_{\min(i+\beta-1,n)}))$ é o comprimento do menor vetor em $L_i(\mathbf{b}_1, \cdots, \mathbf{b}_{\min(i+\beta-1,n)})$.

O valor do parâmetro β influencia diretamente na qualidade das bases obtidas com o BKZ, mas também afeta o tempo de execução. Um valor tipicamente empregado é $\beta = 20$, mas qualquer valor entre 2 e a dimensão do reticulado pode ser utilizado.

Um fato interessante é que o próprio LLL é executado dentro do BKZ, mas este permite obter bases melhores do que o LLL, sobretudo em dimensões maiores. Todavia o seu funcionamento é demasiadamente complexo e envolve detalhes que tornariam este capítulo longo demais. Por esta razão, sua descrição será omitida.

4 OS SISTEMAS GGH E GGH-YK

> "Vivemos todos, neste mundo, a bordo de um navio saído de um porto que desconhecemos para um porto que ignoramos." (Fernando Pessoa)

Neste capítulo serão discutidos os aspectos teóricos e computacionais dos sistemas GGH e GGH-YK, desde a criação de chaves, encriptação e decriptação até o uso da Forma Hermitiana Normal, que é uma forma canônica para representação de bases de reticulados.

4.1 Empregando o CVP como problema subjacente

No Capítulo 1 discutimos o paradigma da criptografia assimétrica e mostramos que o seu funcionamento depende das Funções Trapdoor Seguras, que são fáceis de calcular mas difíceis de inverter, a menos que se conheça alguma informação especial, codificada na chave secreta. Além disso, inverter a FTS sem o conhecimento da chave secreta deve ser tão difícil quanto resolver algum problema computacionalmente intratável, que é dito o *problema subjacente* à FTS.

O GGH consiste em uma FTS cujo problema subjacente é o CVP. A ideia principal deste método é encriptar uma mensagem em um vetor $\mathbf{c} = \mathbf{v} + \mathbf{r}$, em que $\mathbf{v}$ é um vetor de algum reticulado conhecido, e $\mathbf{r}$ é um vetor de perturbação aleatório cuja norma seja relativamente pequena. Desta maneira, $\mathbf{v}$ é o vetor do reticulado mais próximo de $\mathbf{c}$, e pode ser recuperado desde que se conheça uma base boa para o reticulado (a chave secreta).

4.2 Geração de chaves, encriptação e decriptação

O primeiro passo é obter a chave secreta, que deve ser uma base boa de algum reticulado. Lembremos que uma base boa é aquela cujo defeito ortogonal é pequeno. É fácil construir uma base de vetores quase ortogonais, bastando escolher uma da forma

$$B = \gamma I + P, \tag{4.1}$$

em que $\gamma \in \mathbb{Z}$ e P é uma matriz com entradas restritas a algum pequeno conjunto de inteiros, digamos $\{-l, \cdots, -1, 0, 1, \cdots, l\}$ para algum inteiro $l \ll \gamma$. Os autores sugerem $l = 4$ e $\gamma = l\lceil\sqrt{n}\rfloor$, em que n é a dimensão de B.

A partir da chave secreta, obtém-se a chave pública W multiplicando B por uma matriz unimodular aleatória U, com entradas inteiras. Portanto, $W = BU$. Antes de encriptar uma mensagem, esta deve ser codificada em um vetor de inteiros $\mathbf{x}$. O criptograma é dado então por

$$\mathbf{c} := W\mathbf{x} + \mathbf{r}, \tag{4.2}$$

em que $\mathbf{r}$ é sorteado aleatoriamente no momento da encriptação. O vetor $\mathbf{r}$ é denominado *chave efêmera*. É importante assegurar que uma chave efêmera diferente seja usada para cada nova mensagem a ser encriptada. Assim, uma mesma mensagem encriptada duas vezes resultará em criptogramas distintos, já que serão utilizadas chaves efêmeras distintas. Por esta razão, dizemos que o GGH emprega o paradigma da *encriptação probabilística*.

Na decriptação, calculam-se as coordenadas de $\mathbf{c}$ na chave secreta B, obtendo-se o vetor $B^{-1}\mathbf{c} = U\mathbf{x} + B^{-1}\mathbf{r}$. Em seguida, estas coordenadas são arredondadas para o inteiro mais próximo, e o resultado obtido é o vetor $\mathbf{z} = U\mathbf{x}$. Finalmente resolve-se o sistema $W\mathbf{y} = B\mathbf{z}$, cuja solução é $\mathbf{x}$. Note que a decriptação é equivalente a

resolver uma instância do CVP, e o método empregado é essencialmente o algoritmo de Babai, discutido no Capítulo 3.

Observe ainda que, para realizar corretamente a decriptação, impusemos a seguinte hipótese:

$$\lceil B^{-1}\mathbf{c} \rfloor = \lceil U\mathbf{x} + B^{-1}\mathbf{r} \rfloor = U\mathbf{x}.$$

Isto somente é possível se a condição abaixo for satisfeita:

$$\mathbf{e} = \lceil B^{-1}\mathbf{r} \rfloor = \mathbf{0}. \tag{4.3}$$

O vetor $\mathbf{e}$ é dito o *vetor erro de arredondamento*. Intuitivamente, a condição (4.3) nos diz que as coordenadas de $\mathbf{c}$ na base B devem estar suficientemente próximas das coordenadas de $W\mathbf{x}$ nesta mesma base, caso contrário haverá erros de decriptação.

Portanto, é necessário impor alguma restrição sobre as entradas de $\mathbf{r}$, a fim de que a condição (4.3) seja satisfeita. Os autores do GGH sugerem escolher $r_i = \pm\sigma$, em que σ é algum número inteiro pequeno. Observe que a condição (4.3) equivale a

$$|\sum_{j=1}^{n} a_{i,j} r_j| < 0,5,$$

para todo $i = 1, \cdots, n$, em que $A = B^{-1}$. Se denotarmos por φ o valor máximo dentre as normas L_1 das linhas de A, temos que

$$|\sum_{j=1}^{n} a_{i,j} r_j| \leq \varphi\sigma.$$

Portanto, para que a decriptação funcione corretamente, basta fazer $r_i = \pm\sigma$, para todo $i = 1, \cdots, n$, tal que

$$\sigma < \frac{1}{2\varphi}. \tag{4.4}$$

No entanto, a condição (4.4) resulta em valores de σ muito pequenos. Para melhorar a segurança do sistema, é possível obter valores maiores para σ, porém o preço a ser

pago é uma probabilidade de erros na decriptação. Os autores do GGH mostram que, para obter probabilidade de erro menor do que ϵ, basta escolher

$$\sigma \leq \frac{1}{\xi\sqrt{8\ln(2n/\epsilon)}}, \tag{4.5}$$

em que $\xi = \sqrt{n} \times \max\{\text{norma } l_\infty \text{ das linhas de } A\}$. Os autores do GGH apresentam uma estimativa para o valor de σ a partir dos parâmetros $n = 120$ e $\epsilon = 10^{-5}$. Gerando chaves secretas da forma (4.1) com $l = 4$, eles obtêm $\sigma \leq 2,5$, portanto $\sigma = 2$.

Resumimos abaixo todas as etapas do GGH, desde a geração de chaves até a encriptação e decriptação:

1. **Geração de chaves:** construa a chave secreta $B = \gamma I + P \in \mathbb{Z}^{n\times n}$, em que $\gamma = m\lceil\sqrt{n}\rfloor$ e $p_{i,j} \in \{-m, \cdots, m\}$, para algum $m \in \mathbb{Z}$. Obtenha a chave pública W multiplicando B por uma (ou várias) matriz unimodular aleatória U.
2. **Geração de parâmetros:** fixe ϵ e escolha $\sigma \in \mathbb{Z}$ tal que $\sigma \leq \frac{1}{\xi\sqrt{8\ln(2n/\epsilon)}}$, em que $\xi = \sqrt{n} \times \max\{\text{norma } l_\infty \text{ das linhas de } B^{-1}\}$.
3. **Encriptação:** codifique a mensagem em um vetor $\mathbf{x} \in \mathbb{Z}^n$ e sorteie aleatoriamente um vetor $\mathbf{r} \in \mathbb{Z}^n$ tal que $r_i = \pm\sigma$. Calcule o criptograma $\mathbf{c} = W\mathbf{x} + \mathbf{r}$.
4. **Decriptação:** calcule o vetor $\mathbf{u} = \lceil B^{-1}\mathbf{c}\rfloor$ e resolva o sistema $W\mathbf{z} = B\mathbf{u}$.

4.3 Redução de bases e a segurança do GGH

O principal ataque ao GGH consiste em tentar reduzir a chave pública, por intermédio de um algoritmo de redução como o LLL ou o BKZ, e realizar o procedi-

mento de decriptação com a base reduzida obtida. Outra possibilidade consiste em aplicar a técnica de imersão discutida no Capítulo 3.

Os autores do GGH afirmam que os ataques conhecidos devem ser inviáveis para dimensão 150, mas especulam que, usando os melhores algoritmos de redução de base (e com o poder computacional disponível naquela época), este limite subiria para 300.

Nguyen fez o primeiro trabalho de criptoanálise do GGH. Seus resultados foram tão contundentes que o sistema chegou a ser declarado oficialmente morto. Os problemas apontados por Nguyen foram os seguintes: primeiro, toda mensagem encriptada deixa vazar uma parte da mensagem original. A segunda falha surge quando tentamos corrigir a primeira, escolhendo as entradas do vetor $\mathbf{r}$ aleatoriamente no conjunto $\{-\sigma, \cdots, \sigma\}$ em vez de $\{-\sigma, \sigma\}$. De acordo com Nguyen, isto torna o sistema ainda mais frágil, porque à medida que se diminui a norma de $\mathbf{r}$, fica mais fácil resolver a instância do CVP que quebra o sistema.

Usando os algoritmos LLL e BKZ, Nguyen foi capaz de resolver todos os desafios do GGH disponíveis na Internet, exceto em dimensão 400, onde ele apenas foi capaz de revelar bits parciais da mensagem original. A fim de tornar o sistema seguro, seria necessário aumentar demasiadamente a dimensão do reticulado (acima de 400), mas isto, argumenta Nguyen, tornaria o sistema impraticável, porque as chaves teriam mais de 2 MB de tamanho.

Para contornar este problema, em [44] sugeriu-se que a chave pública fosse obtida calculando-se a *Forma Hermitiana Normal* da chave secreta.

Definição 4.1 *Seja $W \in \mathbb{Z}^{n \times n}$ uma base. Dizemos que W está na Forma Hermitiana Normal (FHN) se:*

1. *W é triangular superior;*

2. *$w_{i,i} > 0$, para todo i;*

3. *$0 \leq w_{i,j} < w_{i,i}$, para todo $j > i$.*

Em outras palavras, a FHN é uma matriz triangular superior, cujas entradas são todas não-negativas e a maior entrada de cada linha está situada na diagonal principal.

Algoritmo 8: FHN(B)

Entrada: $B \in \mathbb{Z}^{n \times n}$
Saída: A matriz $W \in \mathbb{Z}^{n \times n}$, correspondente à Forma Hermitiana Normal de B
início
  $W \longleftarrow B$;
  $i \longleftarrow n$;
  enquanto $i \geq 1$ **faça**
    Escolha k tal que $0 < |w_{i,k}| \leq |w_{i,j}|$, para todo $1 \leq j \leq n$;
    $\mathbf{w}_i \longleftrightarrow \mathbf{w}_k$;
    se $w_{i,i} < 0$ **então**
      $\mathbf{w}_i \longleftarrow -\mathbf{w}_i$;
    fim se
    para $j = i-1, \cdots, 1$ **faça**
      se $w_{i,j} < 0$ **então**
        $\mathbf{w}_j \longleftarrow -\mathbf{w}_j$;
      fim se
      enquanto $w_{i,i} > 0$ **faça**
        Calcule q e r tais que $w_{i,j} = qw_{i,i} + r$;
        $\mathbf{w}_j \longleftarrow \mathbf{w}_j - q\mathbf{w}_i$;
        $\mathbf{w}_i \longleftrightarrow \mathbf{w}_j$;
      fim enqto
      $\mathbf{w}_i \longleftrightarrow \mathbf{w}_j$;
      para $j = i+1, \cdots, n$ **faça**
        Calcule q e r tais que $w_{i,j} = qw_{i,i} + r$;
        $\mathbf{w}_j \longleftarrow \mathbf{w}_j - q\mathbf{w}_i$;
      fim para
      $i \longleftarrow i - 1$;
    fim para
  fim enqto
fim

O algoritmo 8 toma como entrada uma matriz quadrada $B \in \mathbb{Z}^{n \times n}$ e retorna a FHN de B. A ideia do algoritmo é a seguinte: para cada linha i da matriz, começando da última, são realizadas operações de troca de colunas, de modo que o elemento não nulo de menor valor absoluto daquela linha fique na diagonal principal. Em seguida, para todo $j < i$, calcula-se o quociente q da divisão inteira entre o

elemento da posição (i, i) e o elemento da posição (i, j). Subtrai-se então q vezes a coluna i da coluna j, e em seguida as colunas i e j são permutadas. Este processo é repetido até que a entrada da posição (i, j) seja igual a 0. Note que, ao final deste processo, o mdc entre os elementos (i, i) e (i, j) iniciais ocupará a posição (i, i).

Em seguida, para todo $j > i$, calcula-se novamente o quociente q da divisão inteira entre o elemento da posição (i, i) e o elemento da posição (i, j). Subtrai-se então q vezes a coluna i da coluna j. Isto garante que o maior elemento de cada linha esteja sempre na diagonal principal. Além disso, antes de cada operação com colunas, certifica-se que as entradas das posições (i, i) e (i, j) sejam positivas. Se não forem, multiplicam-se as colunas i e/ou j por -1.

Observe que são realizadas apenas combinações lineares entre colunas e trocas de sinal. Portanto, o cálculo da FHN preserva o valor absoluto do determinante da matriz original. Assim, se B é uma base e $W = FHN(B)$, temos que B e W geram o mesmo reticulado, o que equivale a dizer que existe uma matriz inteira unimodular U tal que $B = WU$. Além disso, todas as operações garantem que, sendo B uma matriz inteira, sua FHN também será inteira.

O que está por trás do uso da FHN é o ganho em eficiência que ela proporciona. Em [44] foi feita uma estimativa para os tamanhos das chaves públicas (em kB) no formato originalmente proposto e na FHN. Os resultados são reproduzidos na Tabela 4.1, que apresenta também o fator de redução (razão entre o tamanho da base sem a FHN e com a FHN).

Além da redução no tamanho das chaves, a FHN proporciona maior segurança, já que as bases neste formato são mais difíceis de reduzir. No Capítulo 7 faremos uma análise detalhada desses aspectos.

Dimensão	Sem FHN	Com FHN	Fator de Redução
200	330	32	$10,3$
250	620	50	$12,4$
300	990	75	$13,2$
350	1630	100	$16,3$
400	2370	140	$16,9$

Tabela 4.1: Estimativa dos tamanhos das chaves públicas (em kB) no GGH, sem a FHN e com a FHN.

Em 2010, o ataque de Nguyen foi estendido por [45], e o último desafio do GGH em dimensão 400 foi resolvido.

4.4 Novos rumos para o GGH

Em 2012, Yoshino e Kunihiro propuseram uma modificação no GGH para empregar chaves efêmeras maiores do que na versão original. Além disso, com as mudanças sugeridas pelos autores, a decriptação deixaria de ser probabilística.

Esta variante do GGH, que batizamos de GGH-YK em referência aos autores, emprega entradas pequenas no vetor $\mathbf{r}$, como no sistema original, e também entradas grandes. Intuitivamente, isto faz com que o criptograma fique *mais distante* do reticulado.

Vamos denotar novamente por n a dimensão do reticulado. Para gerar a chave secreta do GGH-YK, são escolhidos dois parâmetros $\alpha \in \mathbb{Z}$ e $\beta \in \mathbb{Q}$, e então calcula-se a matriz

$$B = \gamma I + P,$$

em que $\gamma = \alpha\lceil n^{\beta} \rfloor \in \mathbb{Z}$ e P é uma matriz aleatória tal que $p_{i,j} \in \{-1, 0, 1\}$. A chave pública W é obtida calculando-se a FHN da matriz secreta. Yoshino e Kunihiro sugerem escolher $\beta = 1$ ou $\beta = 0,5$. Note que o valor de γ não é publicamente

divulgado.

Em seguida, devem ser escolhidos os parâmetros públicos do sistema, que vamos denotar por (σ, h, k), tal que $\sigma < h$. O parâmetro σ corresponderá ao valor absoluto máximo das entradas pequenas de $\mathbf{r}$, h será o valor absoluto das entradas *grandes* de $\mathbf{r}$ e k a quantidade de entradas grandes. Além disso, vamos denotar por l o comprimento em bits da mensagem a ser codificada, tal que

$$l = \lceil \frac{2n\sigma}{2\sigma + 1} \rfloor, \tag{4.6}$$

e $I_n = \{1, 2, \cdots, n\}$ o conjunto de índices. A ideia é codificar a mensagem nas entradas não nulas do vetor $\mathbf{r}$, atribuindo valores negativos aos bits 0, e valores positivos aos bits 1.

Para codificar uma mensagem binária $m \in \{0,1\}^l$, selecionam-se aleatoriamente dois subconjuntos $S, T \subset I_n$, tais que $\#T = n - l$, $\#S = k$ e $S \cap T = \emptyset$. Para todo $j \in T$, fixamos $r_j = 0$, o que nos deixa com exatamente l índices para codificar os bits da mensagem. A função do Algoritmo 9 retorna os subconjuntos S e T.

Algoritmo 9: seleciona(n, l, k)

Entrada: $n, l, k \in \mathbb{Z}$
Saída: Dois subconjuntos de índices S e T
início
  $I_n \longleftarrow \{1, \cdots, n\}$;
  para $j \longleftarrow 1, \cdots, n - l$ **faça**
    $x \overset{aleat.}{\longleftarrow} y \in I_n$;
    $T \longleftarrow T \cup \{x\}$;
    $I_n \longleftarrow I_n - \{x\}$;
  fim para
  para $j \longleftarrow 1, \cdots, k$ **faça**
    $x \overset{aleat.}{\longleftarrow} y \in I_n$;
    $S \longleftarrow S \cup \{x\}$;
    $I_n \longleftarrow I_n - \{x\}$;
  fim para
  retorna (S, T)
fim

Uma vez selecionados os conjuntos S e T, a mensagem m pode ser codificada. Para isto, dois contadores são iniciados, digamos i e j: o primeiro percorre o

conjunto I_n, enquanto o segundo percorre os índices da mensagem, recebendo valor inicial nulo. Cada vez que $i \notin T$, j é incrementado e o bit m_j é codificado, obedecendo à seguinte regra: se $i \in S$, então $r_i = \pm h$, sendo negativo se $m_j = 0$ e positivo caso contrário. Por outro lado, se $i \notin S$, então r_i recebe aleatoriamente um valor de $\{-\sigma, \cdots, -1\} \cup \{1, \cdots, \sigma\}$, sendo negativo se $m_j = 0$ e positivo caso contrário. A função do Algoritmo 10 codifica a string binária m no vetor $\mathbf{r}$, seguindo o procedimento que acabamos de descrever.

Algoritmo 10: codifica(n, σ, h, k, l, m)

```
Entrada: n, σ, h, k, l ∈ ℤ e m ∈ {0,1}^l
Saída: r ∈ ℤ^n
início
    (S, T) = seleciona(n, l, k);
    j ⟵ 0;
    para i ⟵ 1, ··· , n faça
        se i ∈ T então
            r_i ⟵ 0;
        senão
            j ⟵ j + 1;
            se i ∈ S então
                se m_j = 0 então
                    r_i ⟵ −h;
                senão
                    r_i ⟵ h;
                fim se
            senão
                se m_j = 0 então
                    r_i ⟵(aleat.) {−σ, ··· , −1};
                senão
                    r_i ⟵(aleat.) {1, ··· , σ};
                fim se
            fim se
        fim se
    fim para
    retorna r
fim
```

Como a mensagem foi codificada no vetor $\mathbf{r}$, deve-se escolher aleatoriamente um ponto do reticulado $L(W)$. Na prática, porém, escolhe-se um ponto que dependa do próprio vetor $\mathbf{r}$, resolvendo-se o sistema $W\mathbf{y} = \mathbf{r}$ e calculando-se o vetor $\mathbf{x} = \lfloor \mathbf{y} \rfloor = \lfloor W^{-1}\mathbf{r} \rfloor$. Assim, o ponto do reticulado escolhido é $-W\mathbf{x}$ e o criptograma é dado por

$$\mathbf{c} := \mathbf{r} - W\mathbf{x}. \tag{4.7}$$

Observe que, no cálculo do vetor $\mathbf{x}$, o sistema $W\mathbf{y} = \mathbf{r}$ pode ser resolvido por retrosubstituição, já que W está na FHN.

Na decriptação, calcula-se o vetor $\mathbf{u} = B^{-1}\mathbf{c} - \lceil B^{-1}\mathbf{c} \rfloor$. Note que, como $L(W) = L(B)$, existe uma matriz unimodular U tal que $W = BU$. Logo,

$$\mathbf{u} = B^{-1}\mathbf{r} - U\mathbf{x} - \lceil B^{-1}\mathbf{r} \rfloor + U\mathbf{x},$$

ou seja, $\mathbf{u} = B^{-1}\mathbf{r} - \lceil B^{-1}\mathbf{r} \rfloor$. Multiplicando por B, obtemos a primeira aproximação para o vetor $\mathbf{r}$, que vamos denotar por

$$\mathbf{r}' = B\mathbf{u} = \mathbf{r} - B\lceil B^{-1}\mathbf{r} \rfloor.$$

A etapa final consiste em determinar o vetor erro de arredondamento $\mathbf{e} = \lceil B^{-1}\mathbf{r} \rfloor$. A proposição a seguir estabelece as entradas do vetor $\mathbf{e}$, em função do valor das entradas correspondentes de $\mathbf{r}'$.

Proposição 4.1 *Se $r'_i < -h - k$, então $e_i = 1$; se $r'_i > h + k$, então $e_i = -1$. Caso contrário, $e_i = 0$.*

Deixaremos a demonstração para a próxima seção. Portanto, depois de calculado o vetor $\mathbf{r}'$, utiliza-se a proposição anterior para determinar as entradas do vetor $\mathbf{e}$ e calcula-se o vetor $\mathbf{r}$, fazendo $\mathbf{r} = \mathbf{r}' + B\mathbf{e}$. Observe que o processo de decriptação é totalmente determinístico, diferentemente do GGH original, em que se paga o preço de uma probabilidade de erros de decriptação ϵ para fixar o valor do parâmetro σ. Mas o aspecto mais importante que diferencia o GGH-YK do GGH é o vetor $\mathbf{e}$. Enquanto no GGH temos $\mathbf{e} = \mathbf{0}$, no GGH-YK permite-se que $\mathbf{e}$ tenha entradas não nulas, e ainda assim a decriptação funciona corretamente.

4.5 Detalhes do GGH-YK

Como acabamos de enfatizar, ideia de Yoshino e Kunihiro é permitir entradas não-nulas no vetor de erro $\mathbf{e} = \lceil B^{-1}\mathbf{r} \rfloor$. É razoável supor que tais entradas, se existirem, devam ocorrer nas posições correspondentes às entradas *grandes* do vetor $\mathbf{r}$, que são aquelas indexadas pelos elementos do conjunto S. Elas correspondem a entradas do vetor $B^{-1}\mathbf{r}$ cujo módulo é maior do que $0,5$.

Por outro lado, nas posições indexadas por elementos de $I_n \setminus S$, as entradas de $\mathbf{e}$ devem ser nulas, correspondendo às entradas de $B^{-1}\mathbf{r}$ cujo módulo é menor do que $0,5$. Seja então $\mathbf{w} = B^{-1}\mathbf{r}$. Assim, para todo $i \notin S$, devemos ter $|w_i| < 1/2$. Fazendo $A = (a_{i,j}) = B^{-1}$, temos

$$|w_i| = |a_{i,i}r_i + \sum_{j \neq i} a_{i,j}r_j| \leq |a_{i,i}r_i| + \sum_{j \neq i} |a_{i,j}r_j|.$$

Então

$$|w_i| \leq \sigma|a_{i,i}| + kh\text{max}|a_{i,j}| + \sigma n\text{max}|a_{i,j}|.$$

Vale ressaltar que a desigualdade anterior é bastante *frouxa*. Na última parcela do lado direito poderia ter sido usado $n - k$ em vez de n, mas decidimos apenas reproduzir os cálculos feitos por Yoshino e Kunihiro.

Portanto, é preciso garantir que $\sigma|a_{i,i}| + kh\text{max}|a_{i,j}| + \sigma n\text{max}|a_{i,j}| < 1/2$. Yoshino e Kunihiro sugerem as seguintes cotas superiores para as entradas de A:

$$|a_{i,i}| < \frac{1}{\gamma}, \tag{4.8}$$

$$|a_{i,j}| \leq \frac{2}{\gamma^2}, \text{ para todo } i \neq j. \tag{4.9}$$

Usando estas desigualdades, obtemos

$$\frac{\sigma}{\gamma} + \frac{2kh}{\gamma^2} + \frac{2\sigma n}{\gamma^2} < \frac{1}{2}. \tag{4.10}$$

Por outro lado, para $i \in S$, deveríamos ter $1/2 < |w_i| < 3/2$. Na verdade, os autores garantem apenas que $|w_i| < 3/2$, assegurando a possibilidade de que $e_i \neq 0$ para algum $i \in S$:

$$|w_i| \leq h|a_{i,i}| + h(k-1)\max|a_{i,j}| + \sigma(n-k)\max|a_{i,j}| < \frac{3}{2}.$$

Mais uma vez, utilizando as desigualdades (4.8) e (4.9), obtemos

$$\frac{h-\sigma}{\gamma} + \frac{2h}{\gamma^2} \leq 1. \tag{4.11}$$

Uma última condição é imposta, a fim de que a decriptação funcione corretamente:

$$2(h+k) < \gamma. \tag{4.12}$$

Em suma, a condição (4.10) garante que $e_i = 0$ para todo $i \notin S$, (4.11) nos dá a esperança de que $e_i \neq 0$ para algum $i \in S$, e (4.12) garante a corretude da decriptação.

Agora que já vimos todos estes detalhes, podemos passar à demonstração da Proposição 4.1. Observe que $\mathbf{r}' + B\mathbf{e} = \mathbf{r}$. Portanto,

$$r_i = r'_i + \sum_{j=1}^{n} b_{i,j}e_j. \tag{4.13}$$

Suponha que $r'_i < -h-k$ e $e_i \in \{-1, 0\}$. Portanto, o somatório acima atinge seu valor máximo quando e_i é máximo. Neste caso, $e_i = 0$. Assim temos

$$r_i < -h-k+\sum_{j\neq i} b_{i,j}e_j.$$

Por outro lado, o somatório acima atinge seu valor máximo quando as entradas não nulas $b_{i,j}$ e as entradas não nulas correspondentes e_j possuem o mesmo sinal. Como estamos supondo que $e_i = 0$, restam no máximo k entradas não nulas do vetor $\mathbf{e}$ no somatório acima. Portanto

$$r_i < -h-k+k = -h,$$

o que á absurdo, logo $e_i = 1$. Analogamente, se supusermos que $r'_i > h + k$ e $e_i \in \{0, 1\}$, o somatório em (4.13) atinge seu valor mínimo quando e_i é mínimo, neste caso $e_i = 0$. Logo

$$r_i > h + k + \sum_{j \neq i} b_{i,j} e_j.$$

O somatório acima atinge seu valor mínimo quando as entradas não nulas $b_{i,j}$ e as entradas não nulas correspondentes e_j possuem sinais opostos. Como estamos supondo novamente que $e_i = 0$, restam no máximo k entradas não nulas do vetor $\mathbf{e}$ no somatório acima. Portanto

$$r_i > h + k + k(-1) = h.$$

Novamente, um absurdo. Suponha finalmente que $-h - k \leq r'_i \leq h + k$ e $e_i = 1$. Neste caso, temos

$$r_i = r'_i + \sum_{j=1}^{n} b_{i,j} e_j \geq r'_i + \min\{b_{i,i}\} e_i + \sum_{j \neq i} b_{i,j} e_j.$$

O valor mínimo de r'_i, por hipótese, é $-h - k$, e o de $b_{i,i}$ é $\gamma - 1$. Além disso, o somatório acima atinge seu valor mínimo quando $b_{i,j} e_j < 0$ para todo $j \neq i$. Portanto,

$$r_i \geq -h - k + (\gamma - 1) + (k - 1)(-1) = \gamma - h - 2k.$$

Por (4.12), temos $r_i > h$. Absurdo. De maneira análoga, suponha que $e_i = -1$. Então:

$$r_i \leq r'_i - \min\{b_{i,i}\} + \sum_{j \neq i} b_{i,j} e_j.$$

Desta vez, o valor máximo de r'_i é $h + k$. Por outro lado, o somatório assume seu valor máximo quando $b_{i,j} e_j > 0$ para todo $j \neq i$. Portanto,

$$r_i \leq h + k - (\gamma - 1) + (k - 1)(1) = -\gamma + h + 2k.$$

Novamente, por (4.12), temos $r_i < -h$. Mais um absurdo. portanto, $e_i = 0$.

□

Em suma, temos o seguinte: no GGH original, a condição essencial para que a decriptação funcione corretamente é que $\mathbf{e} = \mathbf{0}$. No GGH-YK, o vetor $\mathbf{e}$ pode ter entradas não nulas, sem que isto afete a decriptação. Além disso, no GGH-YK a decriptação é determinística, fato que é assegurado pela Proposição 4.1.

Quanto às condições impostas sobre os parâmetros do sistema, a condição (4.10) garante que $e_i = 0$, para todo $i \notin S$. Já a desigualdade (4.11) nos dá esperança de que $e_i \neq 0$, para algum $i \in S$, e a condição (4.12) garante a corretude do processo de decriptação.

Resumimos a seguir as etapas de geração de parâmetros, encriptação e decriptação no GGH-YK:

1. **Geração de chaves:** obtenha a chave secreta $B = \gamma I + P \in \mathbb{Z}^{n \times n}$, cuja inversa satisfaça (4.8) e (4.9), com $p_{i,j} \in \{-1, 0, 1\}$ e $\gamma = \alpha \lceil n^{\beta} \rfloor$, para $\alpha \in \mathbb{Z}$ e $\beta \in \{1/2, 1\}$. Calcule a chave pública $W = FHN(B) \in \mathbb{Z}^{n \times n}$.

2. **Geração de parâmetros públicos:** gere os parâmetros públicos (σ, h, k) satisfazendo as condições (4.10), (4.11) e (4.12). Defina l de acordo com (4.6).

3. **Codificação:** selecione os subconjuntos S e T, de acordo com o Algoritmo 9. Codifique a mensagem $m \in \{0,1\}^l$ no vetor $\mathbf{r} \in \mathbb{Z}^n$, de acordo com o Algoritmo 10.

4. **Encriptação:** calcule o criptograma $\mathbf{c} = \mathbf{r} - W\mathbf{x}$, em que $\mathbf{x} = \lfloor W^{-1}\mathbf{r} \rfloor$.

5. **Decriptação:** calcule $\mathbf{u} = B^{-1}\mathbf{c} - \lceil B^{-1}\mathbf{c} \rfloor$. Multiplique por B e obtenha a aproximação $\mathbf{r}' = B\mathbf{u}$. Construa o vetor $\mathbf{e}$. Se $r'_i < -h - k$, faça $e_i = 1$; se $r'_i > h + k$, faça $e_i = -1$. Caso contrário, $e_i = 0$. Obtenha $\mathbf{r} = \mathbf{r}' + B\mathbf{e}$.

Como será discutido no próximo capítulo, na prática nunca ocorre $e_i \neq 0$, de modo

que o GGH-YK ainda pode ser quebrado pela resolução do CVP, desde que se consiga reduzir eficientemente a base pública. Em outras palavras, o GGH-YK não se comporta de maneira diferente do GGH original.

5 GGHYK-M: UMA VARIANTE DO GGH-YK

> "O amor é um misticismo que quer praticar-se, uma impossibilidade que só é sonhada como devendo ser realizada." (Fernando Pessoa)

Neste capítulo mostramos que o GGH-YK não se comporta de maneira diferente do GGH. Além disso, propomos uma nova variante, que será batizada de GGHYK-M (GGH-YK Modificado).

5.1 Melhorias no esquema de Yoshino e Kunihiro

Neste capítulo propomos algumas possíveis melhorias no GGH-YK. Começamos lançando uma objeção em relação à desigualdade (4.8). Os autores afirmam que esta condição, assim como todas as outras, são satisfeitas com *grande probabilidade*, sugerindo que seja razoavelmente *fácil* gerar uma chave secreta cuja inversa satisfaça esta condição. No entanto, em nossos experimentos, verificamos ser muito difícil satisfazer (4.8). Na prática, sempre existe pelo menos um índice i tal que $|a_{i,i}|$ é um pouco maior do que $1/\gamma$.

Verificamos contudo que é fácil satisfazer a condição $|a_{i,i}| < 2/\gamma$. Além disso, por razões que ficarão mais claras um pouco mais adiante, exigimos que $|a_{i,i}| > 1/\gamma$. Assim, para o GGHYK-M, foram impostas as condições abaixo sobre as entradas da inversa da chave secreta:

$$\frac{1}{\gamma} < |a_{i,i}| < \frac{2}{\gamma} \tag{5.1}$$

$$|a_{i,j}| \leq \frac{2}{\gamma^2} \tag{5.2}$$

Após estas modificações, (4.10) e (4.11) também devem ser alteradas. Empregando os mesmos métodos usados para deduzir essas desigualdades, obtemos

$$\frac{2\sigma}{\gamma} + \frac{2kh}{\gamma^2} + \frac{2n\sigma}{\gamma^2} < \frac{1}{2} + \frac{2\sigma(k+1)}{\gamma^2}. \tag{5.3}$$

para todo $i \notin S$. Por outro lado, para $i \in S$, obtemos

$$2(h-\sigma)\left(\frac{1}{\gamma} - \frac{1}{\gamma^2}\right) < 1. \tag{5.4}$$

Substituindo (4.8), (4.10) e (4.11) respectivamente por (5.1), (5.3) e (5.4), mas mantendo (4.9) e (4.12), o sistema ainda se comporta exatamente como o GGH original. Como será discutido a seguir, isto se deve à desigualdade (4.12)

5.2 Construindo uma nova variante

A condição (4.12) impõe uma restrição sobre os valores de h e k. Informalmente, ela nos diz que o vetor $\mathbf{r}$ não pode ter muitas entradas grandes, e essas entradas não podem ser *muito grandes*. Além disso, ela é necessária para que a decriptação no GGH-YK funcione corretamente.

Em nossos experimentos, contudo, verificamos que o vetor $\mathbf{e}$ é sempre o vetor nulo se a condição (4.12) for mantida. Consequentemente, o GGH-YK não é diferente do GGH original, já que a decriptação ainda é equivalente a resolver uma instância do CVP. Isto nos deu uma pista de que as entradas grandes de $\mathbf{r}$ deveriam ser ainda maiores.

Mais uma vez, seja $\mathbf{w} = B^{-1}\mathbf{r}$. Para impedir que $\mathbf{e}$ seja o vetor nulo, devemos garantir que $|w_i| > 1/2$, para todo $i \in S$, uma vez que as condições (4.11) e (5.4) apenas garantem que $|w_i| < 3/2$ para os índices em S. Portanto, uma nova condição

é necessária. Lembremos que

$$|w_i| = |a_{i,i}r_i + \sum_{j \neq i} a_{i,j}r_j|,$$

em que $A = (a_{i,j}) = B^{-1}$. *Supondo que todas as entradas de A e r sejam não negativas*, podemos impor nossa condição, já que neste caso temos

$$|w_i| \geq a_{i,i}r_i \geq \min\{a_{i,i}\}\, h.$$

Pela desigualdade (5.1), temos uma cota inferior para as entradas da diagonal principal de A, de modo que, a fim de garantir que $|w_i| > 1/2$, para $i \in S$, basta fazer

$$\frac{h}{\gamma} > \frac{1}{2}, \tag{5.5}$$

o que implica em $2h > \gamma$, contradizendo (4.12). Observe que o vetor $\mathbf{w}$ agora satisfaz $1/2 < |w_i| < 3/2$, para todo $i \in S$, e $|w_i| < 1/2$ para $i \notin S$. Além disso, como estamos supondo que A e $\mathbf{r}$ têm entradas não negativas, $w_i > 0$, para todo i. Portanto, o vetor $\mathbf{e} = \lceil B^{-1}\mathbf{r} \rfloor$ satisfaz agora as seguintes propriedades:

$$e_i = \begin{cases} 0 & \text{para todo } i \notin S; \\ 1 & \text{para todo } i \in S. \end{cases} \tag{5.6}$$

Além disso, a decriptação no GGH-YK depende de (4.12). Como (5.5) contradiz essa condição, é preciso modificar o método de encriptação para o GGHYK-M. Antes, porém, vamos mostrar como obter a matriz A e o vetor $\mathbf{r}$ com entradas não negativas.

Definição 5.1 *Uma matriz quadrada B é dita inversa-positiva se ela é não-singular, e sua inversa possui apenas entradas não negativas.*

A ideia é construir a chave secreta de modo que ela seja inversa-positiva. Felizmente, existe uma sub-classe de matrizes que serve aos nossos propósitos.

Definição 5.2 *Uma matriz quadrada B é dita uma M-matriz se ela pode ser escrita na forma $B = \gamma I + P$, com $p_{i,j} \leq 0$ e $\gamma \geq \rho(P)$, em que $\rho(P)$ é o raio espectral de P.*

Proposição 5.1 *Toda M-matriz é inversa-positiva.*

Demonstração: esta Proposição será demonstrada na próxima seção.

Com o resultado desta proposição, podemos construir a chave secreta do GGHYK-M inversa-positiva, bastando escolher $p_{i,j} \in \{-1, 0\}$. Além disso, a proposta de Yoshino e Kunihiro para $\gamma = \alpha\lceil n^{\beta} \rfloor$, para valores inteiros de α e $\beta \in \{1/2, 1\}$, é suficiente para assegurar que $\gamma > \rho(P)$. Note que a chave secreta B satisfaz agora as seguintes propriedades:

$$b_{i,j} \in \{-1, 0\}\,, \text{ se } i \neq j; \tag{5.7}$$

$$b_{i,j} \in \{\gamma - 1, \gamma\}\,, \text{ se } i = j. \tag{5.8}$$

Resta desenvolver um método de codificar mensagens binárias em vetores com entradas positivas. Uma simples modificação no esquema proposto por Yoshino e Kunihiro serve aos nossos propósitos.

Lembremos que, no GGH-YK, os bits de uma mensagem de comprimento l são codificados nas entradas não nulas de um vetor $\mathbf{r}$ de tamanho $n \geq l$. As entradas nulas, se existirem, são redundantes. Nossa ideia é eliminar as entradas nulas (fazer $l = n - k$) e transferir a redundância para as entradas grandes (de valor absoluto h). Assim, os bits da mensagem serão codificados nas entradas pequenas, de valor absoluto menor do que σ.

A função descrita no Algoritmo 11 gera o conjunto de índices S, tal que $r_i = h$ para todo $i \in S$. O subconjunto T não é mais utilizado nesta versão modificada.

Algoritmo 11: seleciona2(n, k)

Entrada: $n, k \in \mathbb{Z}$
Saída: Um subconjunto de índices S
início
  $I_n \longleftarrow \{1, \cdots, n\}$;
  para $j \longleftarrow 1, \cdots, k$ **faça**
    $x \overset{aleat.}{\longleftarrow} y \in I_n$;
    $S \longleftarrow S \cup \{x\}$;
    $I_n \longleftarrow I_n - \{x\}$;
  fim para
  retorna S
fim

A ideia agora é codificar bits 0 em entradas menores ou iguais a $\sigma/2$, e bits 1 em entradas maiores do que $\sigma/2$ e menores ou iguais a σ, de acordo com o Algoritmo 12:

Algoritmo 12: codifica2(n, σ, h, k, m)

Entrada: $n, \sigma, h, k \in \mathbb{Z}$ e $m \in \{0,1\}^n$
Saída: $\mathbf{r} \in \mathbb{Z}^n$
início
  $S =$ seleciona2(n, k);
  para $i \longleftarrow 1, \cdots, n$ **faça**
    se $i \in S$ **então**
      $r_i \longleftarrow h$;
    senão
      se $m_i = 0$ **então**
        $r_i \overset{aleat.}{\longleftarrow} \{1, \cdots, \sigma/2\}$;
      senão
        $r_i \overset{aleat.}{\longleftarrow} \{\sigma/2 + 1, \cdots, \sigma\}$;
      fim se
    fim se
  fim para
fim

retorna r

A encriptação de $\mathbf{r}$ será feita da mesma forma que no GGH-YK, ou seja,

$$\mathbf{c} := \mathbf{r} - W\mathbf{x},$$

em que $\mathbf{x} = \lfloor W^{-1}\mathbf{r} \rfloor$. A decriptação de $\mathbf{c}$ também é parecida com a do GGH-YK, com algumas ligeiras modificações: calcula-se o vetor $\mathbf{u} = B^{-1}\mathbf{c} - \lceil B^{-1}\mathbf{c} \rfloor = B^{-1}\mathbf{r} - \lceil B^{-1}\mathbf{r} \rfloor$. Multiplicando por B, obtemos a primeira aproximação para o vetor $\mathbf{r}$, dada por

$$\mathbf{r}' = B\mathbf{u} = \mathbf{r} - B\lceil B^{-1}\mathbf{r} \rfloor.$$

Assim como no GGH-YK, a próxima etapa consiste em determinar as entradas de **e**. O método que empregamos anteriormente deve ser modificado, já que o vetor **e** só possui entradas 0 ou 1.

Seja $i \notin S$. Então

$$r'_i = r_i - \sum_{j=1}^{n} b_{i,j}e_j \leq \sigma - \sum_{j \neq i} b_{i,j}e_j.$$

Note que tomamos $j \neq i$ porque $e_i = 0$. O valor mínimo do somatório acima é 0 e ocorre quando as entradas não nulas de **e** são multiplicadas por entradas $b_{i,j}$ nulas. Por outro lado, o valor máximo se dá quando todas as entradas não nulas de **e** são multiplicadas pelas entradas não nulas $b_{i,j}$. Por (5.6) e (5.7), temos

$$0 \leq -\sum_{j \neq i} b_{i,j}e_j \leq -k(-1)(1) = k.$$

Como as entradas de **r** são estritamente positivas, temos que $r'_i \geq 0$. Portanto, vale a seguinte desigualdade:

$$0 < r'_i \leq \sigma + k, \tag{5.9}$$

para todo $i \notin S$. Similarmente, se $i \in S$, temos que

$$r'_i = r_i - \sum_{j=1}^{n} b_{i,j}e_j \leq h - b_{i,i}e_i - \sum_{j \neq i} b_{i,j}e_j.$$

Diferentemente do caso anterior, $e_i = 1$. Então, por (5.8),

$$-b_{i,i}e_i \leq -(\gamma - 1) = 1 - \gamma.$$

Na soma sobre os índices $j \neq i$, restam apenas $k - 1$ entradas não nulas de **e**, e o valor máximo deste somatório, como no caso anterior, se dá quando todas estas entradas são multiplicadas por entradas $b_{i,j}$ também não nulas. Portanto,

$$-\sum_{j \neq i} b_{i,j}e_j \leq -(k-1)(-1)(1) = k - 1.$$

Assim, para todo $i \in S$ vale a seguinte desigualdade:

$$r'_i \leq h + k - \gamma. \tag{5.10}$$

Seria conveniente forçar $r'_i < 0$ para todo $i \in S$. Assim, ao obtermos o vetor $\mathbf{r}'$, poderíamos distinguir facilmente os índices de S daqueles que não estão em S. Para que isto ocorra, é suficiente impor que

$$h + k < \gamma. \tag{5.11}$$

Assim, para finalizar a decriptação no GGHYK-M, basta usar o fato de que $r'_i > 0$ se $i \notin S$, e $r'_i < 0$ se $i \in S$. Portanto, fazemos $e_i = 1$ se $r'_i < 0$, e $e_i = 0$ caso contrário. Uma vez determinado o vetor $\mathbf{e}$, a mensagem $\mathbf{r}$ pode ser recuperada, já que $\mathbf{r} = \mathbf{r}' + B\mathbf{e}$.

Assim como fizemos no capítulo anterior, resumimos agora as etapas do GGHYK-M:

1. **Geração de chaves:** obtenha a chave secreta $B = \gamma I + P \in \mathbb{Z}^{n\times n}$, cuja inversa satisfaça (5.1) e (5.2), com $p_{i,j} \in \{-1, 0\}$ e $\gamma = \alpha\lceil n^\beta \rfloor$, para $\alpha \in \mathbb{Z}$ e $\beta \in \{1/2, 1\}$. Calcule a chave pública $W = FHN(B) \in \mathbb{Z}^{n\times n}$.

2. **Geração de parâmetros públicos:** gere os parâmetros públicos (σ, h, k) satisfazendo as condições (5.3), (5.4), (5.5) e (5.11).

3. **Codificação:** selecione o subconjunto S, de acordo com o Algoritmo 11. Codifique a mensagem $m \in \{0, 1\}^{n-k}$ no vetor $\mathbf{r} \in \mathbb{Z}^n$, de acordo com o Algoritmo 12.

4. **Encriptação:** calcule o criptograma $\mathbf{c} = \mathbf{r} - W\mathbf{x}$, em que $\mathbf{x} = \lfloor W^{-1}\mathbf{r} \rfloor$.

5. **Decriptação:** calcule $\mathbf{u} = B^{-1}\mathbf{c} - \lceil B^{-1}\mathbf{c} \rfloor$. Multiplique por B e obtenha a aproximação $\mathbf{r}' = B\mathbf{u}$. Construa o vetor $\mathbf{e}$. Se $r'_i < 0$, faça $e_i = 1$, caso contrário faça $e_i = 0$. Obtenha $\mathbf{r} = \mathbf{r}' + B\mathbf{e}$.

5.3 M-matrizes

Nesta seção apresentamos uma breve descrição das M-matrizes, encerrando com a demonstração da Prop. 5.1. Para um tratado completo do assunto, recomenda-se a leitura de [46].

Conforme já foi definido na seção anterior, uma M-matriz pode ser escrita na forma

$$B = \gamma I + P,$$

em que $\gamma \geq \rho(P)$ e $p_{i,j} \leq 0$. Para mostrar que toda M-matriz é inversa-positiva, vamos utilizar o conceito de *matrizes convergentes*.

Definição 5.3 *Uma matriz quadrada $T \in \mathbb{R}^{n\times n}$ é dita convergente se*

$$\lim_{k\to\infty} (T^k)_{i,j} = 0,$$

para cada $1 \leq i, j \leq n$.

Em outras palavras, uma matriz é convergente se suas entradas tendem a zero quando quando ela é elevada a expoentes arbitrariamente grandes. É possível mostrar que esta definição equivale a dizer que o raio espectral de T é menor do que 1.

Lema 5.3.1 *Se uma matriz não-negativa T é convergente, então $(I - T)^{-1}$ existe e também é não negativa.*

Demonstração: suponha que T seja convergente. Note que

$$(I - T)(I + T + \cdots + T^k) = I - T^{k+1}.$$

Fazendo k tender a infinito, temos que

$$(I - T)\sum_{k=0}^{\infty} T^k = I - \lim_{k\to\infty} T^{k+1}.$$

Por hipótese, T é convergente, logo temos

$$(I - T)\sum_{k=0}^{\infty} T^k = I.$$

Ou seja,

$$(I - T)^{-1} = \sum_{k=0}^{\infty} T^k.$$

Como T é não negativa, qualquer potência de T resulta em uma matriz também não negativa. Portanto, $(I - T)^{-1} \geq 0$.

□

Agora podemos demonstrar a Prop. 5.1, que afirma que toda M-matriz é inversa-positiva. Seja então $B = \gamma I - P$ uma M-matriz, em que $p_{i,j} \geq 0$. Seja

$$T = \frac{1}{\gamma}P = I - \frac{1}{\gamma}B.$$

Como $\gamma > \rho(P)$, temos que T é convergente. De fato, seja λ um autovalor de P. Então, temos que $|\lambda| \leq \rho(P)$ e $P\mathbf{v} = \lambda\mathbf{v}$, para algum vetor $\mathbf{v}$. Logo,

$$T\mathbf{v} = \frac{1}{\gamma}P\mathbf{v} = \frac{\lambda}{\gamma}\mathbf{v},$$

ou seja, todo autovalor de T é igual a um autovalor de P dividido por γ. Como $\gamma > \rho(P)$, segue que todo autovalor de T é menor do que 1, de modo que $\rho(T) < 1$. Pelo lema anterior, $(I - T)^{-1}$ é não negativa. Mas

$$I - T = \frac{1}{\gamma}B.$$

Portanto, γB^{-1} é não negativa, e consequentemente B^{-1} é não negativa.

□

6 PROTOCOLOS DE AUTENTICAÇÃO BASEADOS NO GGHYK-M

> "Desejo o que não desejo e abdico do que não tenho. Não posso ser nada nem tudo: sou a ponte de passagem entre o que não tenho e o que não quero." (Fernando Pessoa)

Neste capítulo serão discutidas algumas propostas de protocolos de autenticação baseados no GGHYK-M. A motivação para incluir um capítulo sobre autenticação neste trabalho surgiu, na verdade, a partir de tentativas de encontrar um método de assinatura empregando o GGHYK-M, já que em [47] foi mostrado que toda assinatura baseada no GGH deixa vazar informação parcial sobre a chave secreta.

6.1 Visão geral dos protocolos de autenticação

Em muitas situações, um indivíduo deve comprovar sua identidade a fim de obter acesso a determinados recursos. Por exemplo, para fazer *login* em um sistema ou conta de *email*, fazer um saque em um caixa eletrônico ou acessar uma área restrita de um prédio.

Os chamados *protocolos de autenticação* foram criados com a finalidade de estabelecer rotinas que permitam a um indivíduo comprovar sua identidade, quando diante de situações como as que acabamos de citar. O indivíduo pode fazer isso mostrando que conhece alguma informação secreta (uma senha, por exemplo), que possui alguma coisa (como um crachá de acesso) ou ambas (para fazer operações

em caixas eletrônicos, deve-se possuir o cartão do banco e conhecer a senha de acesso). Em [48] é oferecida uma ampla descrição do funcionamento dos protocolos de autenticação.

Existem duas maneiras de provar que se conhece alguma informação secreta: pode-se efetivamente apresentar esta informação, ou então pode-se resolver algum problema ou desafio que comprovadamente só pode ser resolvido quando se conhece esta informação.

Em termos gerais, um protocolo de autenticação envolve um *provador* e um *verificador* que trocam mensagens entre si. O objetivo final é que o provador consiga, como o próprio nome sugere, provar sua identidade ao verificador. Este, por sua vez, pode aceitar ou rejeitar a afirmação do provador.

Ao longo deste capítulo, vamos usar as seguintes convenções:

- As mensagens serão trocadas entre Alice (provador) e Bernardo (verificador) - ou seja, Alice tentará comprovar sua identidade para Bernardo;
- As mensagens trocadas entre eles serão representadas por setas, indicando o sentido e o conteúdo da mensagem. Por exemplo,

  $$A \longrightarrow B : x$$

  representa a mensagem cujo conteúdo é x, enviada por Alice (A) para Bernardo (B);
- Quando forem enviadas mensagens encriptadas, $P_x(m)$ irá denotar a encriptação da mensagem m com a chave pública de X.
- Para os protocolos baseados no GGHYK-M, a chave secreta será denotada

por B e a pública por W, como de costume. Podem ser adotados índices para distinguir as chaves de Alice das chaves de Bernardo (B_a de B_b e W_a de W_b).

6.2 Autenticação por senhas

Conhecida como *autenticação fraca*, a autenticação por senhas é bastante comum. Nela, o provador deve mostrar que conhece alguma informação secreta, neste caso, uma senha previamente compartilhada entre ele e o verificador.

O exemplo mais comum de aplicação deste modelo é nos ambientes de login. Quando um indivíduo cria uma conta de usuário em um computador, por exemplo, ele cria um par $(\text{NOME}, \text{SENHA})$ que fica armazenado no sistema (verificador). Para requisitar acesso ao sistema posteriormente, ele deve fornecer o seu par $(\text{NOME}, \text{SENHA})$, que é comparado com aquele previamente armazenado. Se o par fornecido coincide com o par armazenado, o acesso é garantido. Caso contrário, o acesso é negado. Neste contexto, o indivíduo está comprovando que é o usuário cuja identidade é dada por "NOME" ao mostrar que conhece a respectiva SENHA.

O problema é que o par $(\text{NOME}, \text{SENHA})$ deve ser armazenado em algum lugar. No caso do sistema operacional, existe um arquivo que contém uma lista com todos os pares relativos a todas as contas de usuário. Para impedir que indivíduos não autorizados tenham acesso a essa lista, tais arquivos devem ser protegidos. É comum o uso de *funções hash* em situações como esta. Uma função hash é como uma função de encriptação irreversível. Não há nenhuma chave secreta que permita recuperar a mensagem a partir do criptograma.

Uma função hash, quando aplicada a uma mensagem m, gera um valor de comprimento fixo denominado *digest* e denotado por $h(m)$. Para que a função seja

considerada segura, deve ser computacionalmente inviável:

1. descobrir qual mensagem gerou um determinado digest;

2. descobrir duas mensagens que resultem no mesmo digest.

Assim, em vez de armazenar diretamente a lista de pares (NOME, SENHA), o sistema armazena os pares (NOME, h(SENHA)). Se um indivíduo não autorizado tiver acesso ao arquivo que contém a lista, ele não será capaz de descobrir as respectivas senhas (desde que a função hash utilizada seja criptograficamente segura).

Quando um usuário solicita acesso ao sistema, ele deve fornecer um par (NOME, SENHA), o sistema calcula o digest da senha e compara com o digest previamente armazenado e associado à identidade do usuário. Se forem coincidentes, o acesso é garantido. Caso contrário, o acesso do usuário é negado.

De modo generalizado, podemos pensar em Alice tentando comprovar sua identidade para Bernardo fornecendo a ele alguma informação secreta previamente compartilhada entre eles.

1. $A \longrightarrow B : (A, s)$

em que A representa a própria identificação de Alice e s é a sua senha. Informalmente, Alice está dizendo: "Eu sou Alice e minha senha é s". Se a senha fornecida por Alice estiver correta, ela é aceita. Caso contrário, é rejeitada. Note que, se o canal de comunicação entre Alice e Bernardo não for seguro, a senha deverá ser encriptada antes de ser enviada.

6.3 Autenticação por *challenge-response*

No método *challenge-response*, Alice deve mostrar que conhece alguma informação secreta *sem revelar esta informação* para Bernardo. Isto pode ser feito da seguinte maneira: Bernardo propõe um desafio, que somente o conhecedor do segredo que Alice alega conhecer seria capaz de solucionar. Ela somente será aceita se conseguir resolver corretamente o desafio. Caso contrário, será rejeitada.

$$\begin{aligned} A &\longleftarrow B: \ desafio \\ A &\longrightarrow B: \ resposta \end{aligned}$$

Uma das maneiras de fazer isso é através da criptografia de chave pública. Bernardo seleciona uma mensagem aleatoriamente e encripta usando a chave pública de Alice. Esta deve provar sua identidade mostrando que é capaz de decriptar a mensagem, ou seja, mostrando que possui a chave secreta sem, no entanto, revelar diretamente esta chave secreta.

$$\begin{aligned} A &\longleftarrow B: \ P_a(r) \\ A &\longrightarrow B: \ r \end{aligned}$$

No esquema acima, Bernardo escolhe aleatoriamente uma mensagem r e a encripta. O resultado é enviado para Alice. Ao receber a mensagem encriptada, Alice a decripta aplicando sua chave secreta, obtendo r', que é enviado de volta para Bernardo. Alice conseguirá comprovar sua identidade desde que $r' = r$.

Na primeira etapa do protocolo, além da encriptação de r, Bernardo envia também um digest de r. Este serve como *testemunha* de que ele realmente conhece o valor de r, evitando assim certos tipos de ataque, em que um adversário seleciona mensagens específicas e tenta extrair informações a partir das respostas obtidas.

6.4 Empregando o GGHYK-M em um protocolo de autenticação

O GGHYK-M pode ser utilizado em um esquema de autenticação, em que Alice comprovaria sua identidade mostrando ter conhecimento de sua chave secreta. O primeiro modo de elaborar o protocolo é tradicional: Bernardo seleciona uma mensagem aleatória m, codifica em um vetor $\mathbf{r}$, encripta usando a chave pública de Alice (W_a) e envia $\mathbf{c} = \mathbf{r} - W_a\mathbf{x}$. Alice então decripta $\mathbf{c}$ e envia $\mathbf{r}$ ou m de volta para Bernardo.

$$\begin{array}{l} A \longleftarrow B : \mathbf{r} - W_a\mathbf{x} \\ A \longrightarrow B : \mathbf{r} \end{array} \tag{6.1}$$

Este método apresenta, contudo, uma potencial vulnerabilidade, já que Eva poderia interceptar as mensagens trocadas entre Alice e Bernardo e, a partir dos diversos pares (mensagem, criptograma) obtidos, tentar extrair alguma informação sobre a chave secreta de Alice.

Para contornar este problema, podemos adotar uma estratégia ligeiramente diferente. Lembremos que, no GGHYK-M, o vetor $\mathbf{e} = \lceil B_a^{-1}\mathbf{r} \rfloor$ satisfaz

$$e_i = \begin{cases} 0, \text{ se } r_i < h; \\ 1, \text{ se } r_i = h. \end{cases}$$

Como $B_a = \gamma I + P$, com $p_{i,j} \in \{-1, 0\}$, as entradas da diagonal principal de B_a são iguais a γ ou $\gamma - 1$, enquanto as entradas fora da diagonal principal são iguais a -1 ou 0. Portanto, o vetor $\mathbf{u} = B_a\mathbf{e}$ satisfaz as seguintes propriedades:

$$\begin{cases} -k \leq u_i \leq 0, \text{ se } r_i < h; \\ \gamma - k + 1 \leq u_i \leq \gamma, \text{ se } r_i = h. \end{cases}$$

Além disso, $\mathbf{u}$ é um vetor do reticulado $L(W)$ e sua norma é consideravelmente pequena. Portanto, $\mathbf{u}$ pode ser enviado como resposta do desafio em vez da mensagem decriptada. Para aceitar a identidade de Alice, Bernardo deve verificar que, de fato, $\mathbf{u} \in L(W)$ e que suas entradas satisfazem as propriedades acima.

$$\begin{array}{l} A \longleftarrow B : \mathbf{r} - W_a\mathbf{x} \\ A \longrightarrow B : B_a\lceil B_a^{-1}\mathbf{r} \rfloor \end{array} \tag{6.2}$$

Observe, contudo, que o valor de γ não é publicamente conhecido, de modo que a segunda desigualdade não pode ser diretamente verificada. Ainda assim, como $\gamma > n$ e $k < n$, temos que $\gamma > k$. Portanto, a condição $u_i > 0$ se $r_i = h$ pode ser testada por Bernardo.

Se Eva tentar se passar por Alice, ao receber a encriptação da mensagem desconhecida $\mathbf{r}$ ela deve ser capaz de encontrar um vetor $\mathbf{v} \in L(W)$ tal que $v_i > 0$ se $r_i = h$ e $v_i \leq 0$ se $r_i < h$. Somente a probabilidade de acertar as posições das entradas grandes e pequenas de $\mathbf{r}$ é dada por

$$\frac{1}{\binom{n}{k}}.$$

Para dar uma ideia de parâmetros reais que empregamos em nossa implementação do GGHYK-M, com $n = 300$ e $k = 32$, esta probabilidade é da ordem de 10^{-43}, ou seja, aproximadamente 2^{-143}. Trata-se de uma chance bastante remota de acertar um palpite ao acaso.

Supondo mesmo assim que Eva seja capaz de escolher um vetor $\mathbf{v}$ adequado, ela ainda deve resolver o sistema $W\mathbf{x} = \mathbf{v}$ e obter solução inteira (caso contrário o vetor escolhido não é um ponto do reticulado). Como, a princípio, o criptograma do GGHYK-M não vaza nenhuma informação a respeito da mensagem, Eva não ganha nenhuma vantagem para escolher o vetor $\mathbf{v}$ a partir das mensagens interceptadas. Ou seja, o seu *palpite* não é melhor do que um chute ao acaso.

O esquema que acabamos de propor é viável sob a hipótese de que o verificador (Bernardo) é honesto, já que com o conhecimento da mensagem $\mathbf{r}$ e do vetor $B_a\lceil B_a^{-1}\mathbf{r}\rfloor$ ele poderia tentar extrair alguma informação da chave secreta de Alice.

6.4.1 Autenticação mútua com o GGHYK-M

Os modelos de autenticação que vimos até agora só permitem *autenticação unilateral* (Alice comprova sua identidade para Bernardo, mas ela não tem nenhuma garantia quanto à identidade dele). Tais modelos partem do princípio de que o verificador é sempre honesto.

O ideal é que haja *autenticação mútua*, em que ambas as partes recebam garantias em relação à identidade da outra. Em outras palavras, tanto Alice quanto Bernardo devem comprovar suas identidades, pois presume-se que nenhum deles seja totalmente honesto *a priori*. No caso de protocolos *challenge-response*, eles devem fazer isso mostrando que conhecem suas respectivas chaves secretas. Isto pode ser feito empregando o GGHYK-M de forma ligeiramente diferente, em que Bernardo codifica uma mensagem binária utilizando um subconjunto S escolhido por Alice, que vamos denotar por S_b. As etapas do protocolo são as seguintes:

1. Alice seleciona um subconjunto $S_b \in I_n$, em que $I_n = \{1, \cdots, n\}$, e o envia encriptado para Bernardo (discutiremos a seguir como fazer isso). O subconjunto S_b contém os índices das entradas *grandes* da mensagem que será posteriormente codificada por Bernardo;

2. Bernardo decripta a mensagem recebida de Alice e recupera S_b, gera uma mensagem aleatória m e codifica $m \in \{0,1\}^{n-k}$, obtendo o vetor $\mathbf{r}$, tal que $r_i = h$ para todo $i \in S_b$. O vetor $\mathbf{r}$ é encriptado com a chave pública GGHYK-M de Alice;

3. Bernardo envia o desafio $\mathbf{c}_1 = \mathbf{r} - W_a\mathbf{x}_1$ para Alice;

4. Alice decripta $\mathbf{c}$ e verifica se a mensagem obtida está corretamente codificada (com as entradas grandes indexadas pelos elementos do conjunto S_b que ela

escolhera previamente). Se a codificação estiver errada, ela aborta a operação;

5. Se a codificação estiver correta, ela encripta $\mathbf{r}$ com a chave pública GGHYK-M de Bernardo e envia $\mathbf{c}_2 = \mathbf{r} - W_b\mathbf{x}_2$;

6. Bernardo decripta $\mathbf{c}_2$ e compara o resultado obtido $\mathbf{r}_o$ com $\mathbf{r}$, aceitando a identidade de Alice se $\mathbf{r}_o = \mathbf{r}$.

Note que todas as mensagens trocadas entre Alice e Bernardo são encriptadas. Se Eva tentar interceptá-las, ela não será capaz de extrair nenhuma informação a respeito das chaves secretas de Alice e Bernardo, desde que as respectivas chaves públicas sejam seguras de fato.

Observe também que, na etapa 4 do protocolo, Alice recebe alguma garantia quanto à identidade de Bernardo, já que somente ele seria capaz de codificar a mensagem $\mathbf{r}$ com os parâmetros corretos. Se Eva tentasse se passar por Bernardo, teria de adivinhar o conjunto S_b escolhido por Alice na etapa 1. Como S tem k elementos, a probabilidade de acerto é dada por

$$\frac{1}{\binom{n}{k}}.$$

Já estimamos anteriormente esta probabilidade para $n = 300$ e $k = 32$. A tarefa de Eva fica ainda mais difícil se empregarmos $k = 150$, porque neste caso a probabilidade acima é da ordem de 10^{-89}, ou seja, cerca de 2^{-296}.

O último aspecto a ser discutido é a encriptação do conjunto S_b na etapa 1. Para fazer isso, Alice escolhe seu próprio subconjunto S_a e codifica uma mensagem aleatória $\mathbf{r}'$, de modo que $r'_i = h$ para todo $i \in S_a$. O conjunto S_b é codificado então no vetor $\mathbf{x}'$, bastando fazer $x'_i = h$ para todo $i \in S_b$. Ao decriptar a primeira mensagem de Alice e recuperar $\mathbf{r}'$, Bernardo consegue obter também o vetor $\mathbf{x}'$, que revelará os índices do conjunto S_b.

6.5 Considerações finais sobre autenticação

Conforme enfatizamos no início, este capítulo surgiu depois de tentativas de elaborar um método de assinatura digital com o GGHYK-M. As propostas de esquemas de autenticação que fizemos aqui estão em fase preliminar e devem passar por modificações em trabalhos futuros.

Um tema de pesquisa a ser explorado daqui para frente é o de protocolos de *conhecimento zero*, baseados no conceito de prova de conhecimento zero, apresentado em [49]. Trata-se de um modelo de protocolo em que o provador não fornece ao verificador nenhuma informação além da comprovação de sua identidade. Os protocolos que propusemos aqui, basedos no método *challenge-response*, não possuem necessariamente esta propriedade, uma vez que o provador fornece informações que não poderiam ser calculadas pelo verificador apenas com o conhecimento das informações públicas. Em outras palavras, a participação do provador resulta em um acréscimo de informações disponíveis. Sendo assim, nada nos garante que alguma informação parcial a respeito do segredo conhecido por ele não possa ser obtida.

7 RESULTADOS, CONCLUSÕES E TRABALHOS FUTUROS

> "Dói viver, mas é de longe. Sentir não importa." (Fernando Pessoa)

Neste capítulo são apresentados os resultados de diversas simulações numéricas, realizadas com o intuito de ilustrar tudo que foi apresentado até aqui. Serão analisados os seguintes tópicos: a eficiência dos algoritmos de redução, a influência da Forma Hermitiana Normal na segurança e comportamento do GGH, GGH-YK e do GGHYK-M.

Todos os experimentos foram realizados com a biblioteca NTL para C++ (versão 5.5.1)[1], criada por V. Shoup, rodando em um processador AMD E-350 dual core, com 1.6 GHz e 4GB de RAM.

7.1 Eficiência dos algoritmos de redução

Conforme já foi discutido previamente, a decriptação no GGH consiste em resolver uma instância do CVP, usando o fato de que $\lceil B^{-1}\mathbf{r} \rfloor = \mathbf{0}$, onde B é a chave secreta e $\mathbf{r}$ a chave efêmera. Portanto, o ataque mais simples contra o GGH consiste em reduzir a chave pública, obter uma base B' e com ela tentar realizar a decriptação. Dado então o criptograma $\mathbf{c} = W\mathbf{x} + \mathbf{r}$, o ataque consiste nas seguintes etapas:

[1] Disponível para download em http://www.shoup.net/ntl/

1. Aplicando redução de base à chave pública W, obtenha a matriz B'.

2. Calcule o vetor $\mathbf{u} = \lceil B'^{-1}\mathbf{c} \rfloor = U'\mathbf{x} + \lceil B'^{-1}\mathbf{r} \rfloor = U'\mathbf{x}$.

3. Resolva o sistema $W\mathbf{z} = B'\mathbf{u}$.

Este tipo de ataque funciona ainda que a base B' seja diferente da base secreta B. A única condição exigida é que B' seja boa o bastante a fim de satisfazer $\lceil B'^{-1}\mathbf{r} \rfloor = \mathbf{0}$. No caso do GGH-YK, isto deixaria de ser verdade. Acontece que, na prática, a chave secreta do GGH-YK sempre satisfaz $\lceil B^{-1}\mathbf{r} \rfloor = \mathbf{0}$. Portanto, se obtivermos uma base B' tal que $\lceil B'^{-1}\mathbf{r} \rfloor = \mathbf{0}$, o ataque que acabamos de descrever funciona também contra o GGH-YK.

Contra o GGHYK-M, o ataque pelo CVP não funciona, a menos que se consiga recuperar a própria base secreta através da redução da chave pública. Já que a base secreta B satisfaz $\lceil B^{-1}\mathbf{r} \rfloor \neq \mathbf{0}$, nenhuma base B' satisfazendo $\lceil B'^{-1}\mathbf{r} \rfloor = \mathbf{0}$ poderá ser obtida através da redução da chave pública. Assim, a única maneira do ataque funcionar é se $B' = B$. Se isto for possível, basta executar o algoritmo de decriptação normalmente e recuperar a mensagem.

Mesmo o ataque por imersão, discutido no Cap. 3, depende da redução da chave pública. Portanto, para medir a segurança do GGH-YK e do GGHYK-M, deve-se medir essencialmente a eficiência dos algoritmos de redução de base. Nesta seção, foi analisado o desempenho dos algoritmos LLL e BKZ contra as chaves públicas do GGH-YK e GGHYK-M, levando-se em conta a qualidade das bases reduzidas obtidas e o tempo de execução. A qualidade das bases foi calculada pela Razão de Hadamard, utilizando-se as siglas RHCP para *Razão de Hadamard da chave pública* e RHBR para *Razão de Hadamard da base reduzida*.

Foram geradas chaves secretas do GGH-YK, com $\gamma = 3n$ e, a partir delas, as respectivas chaves públicas. Aplicou-se então o LLL e o BKZ, medindo-se a qualidade das bases obtidas e o tempo de execução. Os resultados estão resumidos nas Tabelas 7.1 e 7.2.

Dimensão	RHCP	RHBR	Tempo de execução (s)
64	$0,4067 \times 10^{-143}$	$0,9994$	$1,59$
128	$0,1819 \times 10^{-327}$	$0,2613$	$45,29$
200	$0,7920 \times 10^{-552}$	$0,1454$	$428,33$
250	$0,3661 \times 10^{-715}$	$0,0992$	$1432,64$
300	$0,2436 \times 10^{-882}$	-	-

Tabela 7.1: Desempenho do LLL contra as chaves públicas do GGH-YK.

Dimensão	RHCP	RHBR	Tempo de execução (s)
64	$0,6837 \times 10^{-143}$	$0,9994$	$1,63$
128	$0,1819 \times 10^{-327}$	$0,9997$	$47,51$
200	$0,7920 \times 10^{-552}$	$0,9998$	$554,03$
250	$0,3661 \times 10^{-715}$	$0,9998$	$5130,86$
300	$0,2436 \times 10^{-882}$	-	-

Tabela 7.2: Desempenho do BKZ contra as chaves públicas do GGH-YK.

Em seguida, foram geradas chaves secretas do GGHYK-M e, a partir delas, as respectivas chaves públicas. Aplicou-se a cada chave pública o LLL e o BKZ, medindo-se a qualidade das bases obtidas e o tempo de execução. Os resultados estão resumidos nas Tabelas 7.3 e 7.4.

Dimensão	RHCP	RHBR	Tempo de execução (s)
64	$0,8299 \times 10^{-143}$	$0,9993$	$1,47$
128	$0,2046 \times 10^{-327}$	$0,9997$	$38,44$
200	$0,2143 \times 10^{-551}$	$0,1546$	$379,26$
250	$0,8855 \times 10^{-715}$	$0,1224$	$1245,93$
300	$0,2933 \times 10^{-882}$	-	-

Tabela 7.3: Desempenho do LLL contra as chaves públicas do GGHYK-M.

Em dimensões acima de 200, o LLL não consegue recuperar bases muito boas, sendo portanto o BKZ mais adequado para trabalhar em dimensões maiores.

É notável que a FHN proporciona bases com Razão de Hadamard extremamente pequena, o que dificulta ainda mais a redução. A partir de dimensão 300, tanto o LLL quanto o BKZ não foram capazes de retornar uma base reduzida. Ambos exibiram uma mensagem durante a execução, informando que não poderiam proceder, e a redução foi interrompida. Também vale ressaltar que as chaves públicas

Dimensão	RHCP	RHBR	Tempo de execução (s)
64	$0,8299 \times 10^{-143}$	$0,9993$	$1,54$
128	$0,2046 \times 10^{-327}$	$0,9997$	$44,6$
200	$0,2143 \times 10^{-551}$	$0,9998$	$453,46$
250	$0,8855 \times 10^{-715}$	$0,9998$	$2984,47$
300	$0,2933 \times 10^{-882}$	-	-

Tabela 7.4: Desempenho do BKZ contra as chaves públicas do GGHYK-M.

do GGHYK-M, obtidas através do cálculo da FHN de M-matrizes, não oferecem a um adversário nenhuma *vantagem* significativa em relação às chaves públicas do GGH-YK (com exceção do tempo de execução dos algoritmos de redução, que foi um pouco menor contra o GGHYK-M, mas a qualidade das bases obtidas foi a mesma). Isto ocorre por duas razões: primeiro, a FHN de M-matrizes não apresenta nenhuma característica que permita distinguí-la da FHN de matrizes genéricas; em segundo lugar, o fato da FHN ter sido calculada a partir de uma M-matriz não *ajuda* de maneira alguma os algoritmos de redução.

A partir dos resultados obtidos, pode-se concluir que o GGHYK-M exige dimensões acima de 300. Como veremos na próxima seção, com o uso da FHN isto não representa um problema, já que as chaves, mesmo em dimensões maiores, possuem tamanhos viáveis na prática.

7.2 Tamanho das bases na Forma Hermitiana Normal

Foi feita uma comparação entre tamanho das chaves públicas do GGH na FHN com o de chaves públicas obtidas pela multiplicação da chave secreta por uma unimodular aleatória, a fim de reproduzir os dados apresentados no Capítulo 4. Os resultados são resumidos na Tabela 7.5.

Dimensão	Sem FHN	Com FHN	Fator de Redução
200	544	27	$20,14$
250	852	46	$18,52$
300	1231	68	$18,1$
350	1685	94	$17,92$
400	2213	124	$17,85$

Tabela 7.5: Comparação entre os tamanhos das chaves públicas (em kB) no GGH, sem a FHN e com a FHN.

Pode-se constatar que nossos resultados confirmam as estimativas apresentadas no Capítulo 4, revelando que a FHN, de fato, proporciona uma redução significativa no tamanho das chaves. Vale ressaltar que os cálculos apresentados aqui representam estimativas extremamente otimistas, uma vez que *desprezamos* as entradas nulas das chaves. Na realidade, mesmo as entradas nulas ocupam espaço na memória.

Em seguida, foi calculado o tamanho das chaves públicas do GGHYK-M, com $\gamma = \alpha\lceil n^{\beta} \rfloor$, para $\alpha = 3$ e $\beta = 1$ (Tabela 7.6).

Dimensão	Tamanho da Chave Pública (kB)
200	45
250	73
300	108
350	151
400	200

Tabela 7.6: Tamanho das chaves públicas do GGHYK-M.

Os resultados mostram que mesmo as chaves de dimensões maiores possuem tamanhos perfeitamente aceitáveis. Novamente ressaltamos o fato de que essas estimativas são otimistas, já que não foi levado em conta o espaço ocupado por entradas nulas. Ainda assim, o tamanho das chaves pode ser considerado bastante viável.

Como as bases na FHN são muito esparsas, podem ser utilizadas estruturas de dados especiais para armazená-las. Esta seria uma forma de aproximar o tamanho real de armazenamento das estimativas apresentadas aqui. Por exemplo, uma matriz na FHN poderia ser armazenada em um vetor contendo apenas as entradas não nulas e suas respectivas posições, evitando assim o armazenamento das entradas nulas.

7.3 Desempenho do GGHYK-M

Nesta seção foi avaliado o desempenho do GGHYK-M nas operações de geração de chaves, encriptação e decriptação. A Tabela 7.7 apresenta o tempo (em segundos) de geração das chaves.

(n, σ, h, k)	Chave Secreta (s)	Chave Pública (s)
$(128, 32, 193, 16)$	$2,36$	$9,28$
$(200, 64, 301, 32)$	$11,93$	$89,94$
$(256, 64, 385, 32)$	$13,86$	$331,53$
$(300, 128, 451, 50)$	$22,29$	$717,54$
$(350, 256, 526, 64)$	$35,45$	$1627,1$
$(400, 256, 601, 64)$	$51,46$	$3075,71$

Tabela 7.7: Tempo em segundos da geração de chaves no GGHYK-M.

Na geração da chave secreta foi levado em conta o tempo de verificação das condições (5.1) e (5.2), enquanto na geração da chave pública foi calculado essencialmente o tempo de cálculo da FHN.

A fim de evitar erros de precisão na decriptação, empregamos apenas aritmé-

tica de números inteiros na implementação do GGHYK-M. Para fazer isso, o cálculo da inversa da chave secreta foi alterado. Em vez de calcular diretamente a matriz $B^{-1} \in \mathbb{R}^{n \times n}$, foi calculada a matriz $A = (\det(B))B^{-1} \in \mathbb{Z}^{n \times n}$.

Com o cálculo da inversa inteira, o processo de decriptação sofreu uma ligeira modificação. Seja então $\mathbf{c}$ o criptograma e $A = (\det(B))B^{-1}$:

1. Calcule os vetores $\mathbf{u} = \lceil \frac{1}{\det(B)} A\mathbf{c} \rfloor$ e $\mathbf{r}' = \mathbf{c} - B\mathbf{u}$.
2. Construa o vetor $\mathbf{e}$. Se $r'_i < 0$, faça $e_i = 1$. Caso contrário, $e_i = 0$. Obtenha $\mathbf{r} = \mathbf{r}' + B\mathbf{e}$.

Na Tabela 7.8 são apresentados os tempos de encriptação e decriptação. A inversão da chave secreta acaba tornando o processo de decriptação um pouco lento.

(n, σ, h, k)	Encriptação (s)	Decriptação (s)
$(128, 32, 193, 16)$	$0,02$	$2,32$
$(200, 64, 301, 32)$	$0,02$	$11,83$
$(256, 64, 385, 32)$	$0,03$	$33,90$
$(300, 128, 451, 50)$	$0,04$	$61,58$
$(350, 256, 526, 64)$	$0,06$	$115,92$
$(400, 256, 601, 64)$	$0,07$	$210,41$

Tabela 7.8: Tempo em segundos de encriptação e decriptação no GGHYK-M.

Para reduzir o tempo de decriptação, pode-se armazenar previamente a inversa inteira da chave secreta, juntamente com o seu determinante. Isto poderia representar uma desvantagem, já que a inversa inteira em geral ocupa muito espaço (em alguns dos nossos experimentos, ela chegou a mais de 70 MB). No entanto, o seu armazenamento prévio se traduz em um ganho substancial no tempo de decriptação, como pode ser visto na Tabela 7.9.

Finalmente na Tabela 7.10 são comparados os tamanhos, em bits, das men-

(n, σ, h, k)	Decriptação (s)
$(128, 32, 193, 16)$	$0,01$
$(200, 64, 301, 32)$	$0,04$
$(256, 64, 385, 32)$	$0,08$
$(300, 128, 451, 50)$	$0,11$
$(350, 256, 526, 64)$	$0,17$
$(400, 256, 601, 64)$	$0,27$

Tabela 7.9: Tempo em segundos da decriptação no GGHYK-M, com a inversa da chave secreta previamente armazenada.

sagens e dos respectivos criptogramas. Percebe-se uma elevada taxa de expansão, já que na maioria dos experimentos o criptograma tornava-se mais de 10 vezes maior do que a mensagem original. Vale ressaltar que a codificação foi feita bit a bit, o que acaba contribuindo para esse aumento, uma vez que cada entrada do vetor $\mathbf{r}$ codifica somente um bit, quando poderia certamente codificar vários bits. Além disso, o vetor $\mathbf{r}$ possui entradas redundantes (as entradas *grandes*), que não codificam nenhum bit da mensagem. O uso de outras técnicas de codificação poderia reduzir essa taxa de expansão, contribuindo assim para aumentar a eficiência do sistema.

(n, σ, h, k)	Mensagem (bits)	Criptograma (bits)
$(128, 32, 193, 16)$	112	1097
$(200, 64, 301, 32)$	168	1841
$(256, 64, 385, 32)$	224	2452
$(300, 128, 451, 50)$	250	2943
$(350, 256, 526, 64)$	286	3511
$(400, 256, 601, 64)$	346	4089

Tabela 7.10: Comparação entre o tamanho em bits das mensagens e dos respectivos criptogramas no GGHYK-M.

7.4 GGH-YK e GGHYK-M

Nesta seção mostramos um exemplo do comportamento do GGH-YK e do GGHYK-M em relação ao vetor erro de arredondamento $\mathbf{e} = \lceil B^{-1}\mathbf{r} \rfloor$, em que B é

a chave secreta e $\mathbf{r}$ é a mensagem codificada. Foram usadas dimensões pequenas, para que os resultados pudessem ser exibidos. Foram obtidos os seguintes resultados para o GGH-YK:

- Parâmetros: $n = 16$, $\sigma = 3$, $h = 8$, $k = 4$ e $l = 14$.
- $S = \{9, 10, 14, 15\}$.
- $T = \{1, 8\}$.
- $\mathbf{r}^T = (0, -2, -2, -1, 3, -2, 1, 0, -8, 8, -1, 2, -3, 8, -8, 2)$.
- $\mathbf{e}^T = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$.

Para o GGHYK-M, os resultados foram os seguintes:

- Parâmetros: $n = 16$, $\sigma = 8$, $h = 25$ e $k = 5$.
- $S = \{1, 2, 3, 6, 16\}$.
- $\mathbf{r}^T = (25, 25, 25, 3, 1, 25, 7, 8, 6, 5, 7, 2, 8, 2, 3, 25)$.
- $\mathbf{e}^T = (1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1)$.

Este exemplo, apesar de pequeno, reflete o comportamento geral do GGH-YK em várias dimensões: o vetor erro de arredondamento é sempre o vetor nulo, exatamente como no GGH. Por outro lado, no GGHYK-M, o vetor $\mathbf{e}$ possui entradas não nulas exatamente nas posições indicadas pelo conjunto S.

Vale ressaltar que, no exemplo acima, não utilizamos os mesmos conjuntos de parâmetros. Isto é perfeitamente compreensível se levarmos em conta que os

parâmetros do GGHYK-M devem satisfazer condições diferentes daqueles do GGH-YK e, portanto, não podem assumir os mesmos valores em dimensões iguais.

7.5 Trabalhos futuros

Neste trabalho foram propostas algumas modificações no GGH-YK, de modo a torná-lo efetivamente diferente do GGH. Tais modificações deram origem a uma nova variante, que batizamos de GGHYK-M.

A segurança do GGHYK-M, assim como do GGH-YK, ainda depende essencialmente do uso de matrizes razoavelmente grandes (dimensões acima de 300) e da dificuldade computacional de reduzir bases na FHN. De qualquer forma, o GGHYK-M garante um comportamento diferente do GGH, enquanto o GGH-YK, na prática, funciona exatamente como o sistema original.

Algumas questões permanecem em aberto e serão exploradas em trabalhos futuros: a possibilidade de empregar dimensões menores, o uso de técnicas para reduzir a taxa de expansão na encriptação, sistemas baseados em reticulados com métodos inerentes (e seguros) de assinatura digital, protocolos de *conhecimento-zero* para autenticação baseados no GGHYK-M ou alguma outra variante do GGH, entre outras.

REFERÊNCIAS

[1] KAHN, D. **The codebreakers** - the story of secret writing. New York: Macmillan, 1967.

[2] SORKIN, A. Lucifer: a cryptographic algorithm. **Cryptologia**, Laguna Hills, CA, v. 8, n. 1, p. 22–42, 1984.

[3] DAEMEN, J. ; RIJMEN, V. **The design of Rijndael:** AES - the advanced encryption standard. Berlin: Springer-Verlag, 2002. (Information Security and Cryptography).

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

[5] HOFFSTEIN, J. ; PIPHER, J. C. ; SILVERMAN, J. H. **An introduction to mathematical cryptography**. New York: Springer, 2008. (Undergraduate Texts in Mathematics).

[6] RIVEST, R. ; SHAMIR, A. ; ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. **Communications of the ACM**, New York, v. 21, n. 2, p. 120–126, Feb. 1978.

[7] COUTINHO, S. C. **Números inteiros e criptografia RSA**. 2. .ed. Rio de Janeiro: IMPA, 2013. (Série de Computação e Matemática).

[8] MILLER, V. S. Use of elliptic curves in cryptography. In: WILLIAMS, H. C. **Advances in cryptology** – CRYPTO'85 Proceedings. Berlin: Springer-Verlag, 1986. p.417–426. (Lecture Notes in Computer Science, v.218).

[9] KOBLITZ, N. Elliptic curve cryptosystems. **Mathematics of Computation**, Providence, v. 48, n. 177, p. 203–209, 1987.

[10] KAYE, P. ; LAFLAMME, R. ; MOSCA, M. **An introduction to quantum computing**. Oxford: Oxford University Press, 2007.

[11] NIELSEN, M. A. ; CHUANG, I. L. **Quantum computation and quantum information**. Cambridge: Cambridge University Press, 2000.

[12] PORTUGAL, R. ; LAVOR, C. C. ; CARVALHO, L. M. ; MACULAN, N. **Uma introdução à computação quântica**. São Carlos: SBMAC, 2004.

[13] SHOR, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. **SIAM Journal on Computing**, Philadelphia, v. 26, n. 5, p. 1484–1509, Oct. 1997.

[14] MCELIECE, R. J. **A public-key cryptosystem based on algebraic number theory**. [S.l.]: Deep Space Network -Jet Propulsion Laboratory, 1978. (DSN Progress Report, 42-44).

[15] GOLDREICH, O. ; GOLDWASSER, S. ; HALEVI, S. Public-key cryptosystems from lattice reduction problems. In: KALISKI, B. S. **Advances in cryptology - Crypto'97**. Proceedings of the 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997. Berlin: Springer,-Verlag, 1997. p.112–131. (Lecture Notes in Computer Science, v. 1294).

[16] AJTAI, M. ; DWORK, C. A public-key cryptosystem with worst-case/average-case equivalence. In: ANNUAL ACM SYMPOSIUM ON THE THEORY OF COMPUTING, 29., 1997, El Paso, TX. **Proceedings ...** New York: ACM, 1997. p.284–293.

[17] NGUYEN, P. ; STERN, J. Cryptanalysis of the Ajtai-Dwork cryptosystem. In: KRAWCZYK, H. **Advances in Cryptology - CRYPTO'98**. Proceedings of the 18th Annual International Cryptology Conference Santa Barbara, California, USA, August 23–27, 1998. Berlin: Springer-Verlag, 1998. p.223–242. (Lecture Notes in Computer Science, v.1462).

[18] CAI, J-Y. ; CUSICK, T. W. A lattice-based public-key cryptosystem. In: TAVARES, S. : MEIJER, H. (Eds.). **Selected areas in cryptography**. Proceedings of the 5 th Annual International Workshop SAC'98 Kingston, Ontario Canada, 1998. Berlin: Springer, 1999. p.219–233. (Lecture Notes in Computer Science, v.1556).

[19] GOLDREICH, O.; GOLDWASSER, S.; HALEVI, S. Eliminating decryption errors in the Ajtai-Dwork cryptosystem. In: KALISKI, B. S. **Advances in cryptology – Crypto'97**. Proceedings of the 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997. Berlin: Springer,-Verlag, 1997. p.105–111. (Lecture Notes in Computer Science, v. 1294).

[20] KAWACHI, A. ; TANAKA, K. ; XAGAWA, K. Multi-bit cryptosystems based on lattice problems. In: OKAMOTO, T. ; WANG, X. (Eds). **Public Key Cryptography - PKC 2007**. Proceedings of the 10th International Conference on Practice and Theory in Public-Key Cryptography, Beijing, China, April 16-20, 2007. Berlin: Springer-Verlag, 2007. p.315–329. (Lecture Notes in Computer Science, v.4450).

[21] AJTAI, M. Representing hard lattices with O(n log n) bits. In: ANNUAL ACM SYMPOSIUM ON THEORY OF COMPUTING, 37., 2005, Baltimore. **Proceedings ...** New York: ACM, 2005. p.94–103.

[22] REGEV, O. Improved inapproximability of lattice and coding problems with preprocessing. In: IEEE ANNUAL CONFERENCE ON COMPUTATIONAL COMPLEXITY, 18., 2003, Aarhus, Denmark. **Proceedings ...** Los Alamitos, IEEE, 2003. p.363–370.

[23] REGEV, O. On lattices, learning with errors, random linear codes, and cryptography. In: STOC'05 – ANNUAL ACM SYMPOSIUM ON THEORY COMPUTING , 37., Baltimore. **Proceedings ...** New York: ACM, 2005. p. 84–93.

[24] NGUYEN, P. Cryptanalysis of the Goldreich-Goldwasser-Halevi cryptosystem from Crypto '97. In: WIENER, M. **Advances in cryptology - Crypto'99**. Proceedings of the 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999. Berlin: Springer-Verlag, 1999. p.288–304. (Lecture Notes in Computer Science, v.1666).

[25] GOLDREICH, O. **Private communication**. [S.l.: s.n.], Jan. 1999.

[26] HOFFSTEIN, J. ; PIPHER, J. C. ; SILVERMAN, J. H. NTRU: a ring-based public key cryptosystem. In: BUHLER, J. P. (Ed). **Algorithmic number theory**. Proceedings of the Third International Symposiun, ANTS-III Portland, Oregon, USA, June 21–25, 1998 Berlin: Springer-Verlag, 1998. p.267–288. (Lecture Notes in Computer Science, v.1423).

[27] YOSHINO, M. ; KUNIHIRO, N. Improving GGH cryptosystem for large error vector. In: INTERNATIONAL SYMPOSIUM ON INFORMATION THEORY AND ITS APPLICATIONS, 2012. Honolulu. **Proceedings ...** Los Alamitos: IEEE, 2012, p.416–420.

[28] HOFFMAN, K. ; KUNZE, R. **Linear algebra**. 2. ed. New York: Pearson, 1971.

[29] LANG, S. **Linear algebra**. Berlin: Springer-Verlag, 2004. (Undergraduate Texts in Mathematics).

[30] WUBBEN, D. ; SEETHALER, D. ; JALDéN, J. ; MATZ, G. Lattice reduction: a survey with applications in wireless communications. **IEEE Signal Processing Magazine**, Piscataway, v. 28, n. 3, p. 70–91, May 2011.

[31] CASSELS, J. W. S. **An introduction to the geometry of numbers**. Berlin: Springer-Verlag, 1997. (Springer Classics in Mathematics).

[32] MICCIANCIO, D. The shortest vector problem is NP-hard to approximate to within some constant. **SIAM Journal on Computing**, Philadelphia, v. 30, n. 6, p. 2008–2035, Mar. 2001.

[33] BOAS, P. V. E. **Another NP-complete problem and the complexity of computing short vectors in a lattice**. Amsterdam: Mathematische Instituut, University of Amsterdam, 1981. (Technical Report 81-04).

[34] BABAI, L. On Lovász' lattice reduction and the nearest lattice point problem. **Combinatorica**, New York, v.6, n.1, p.1–13, 1986.

[35] KANNAN, R. Minkowski's convex body theorem and integer programming. **Mathematics of Operations Research**, Hanover, v. 12, n. 3, p. 415–440, Aug. 1987.

[36] LENSTRA, A. K. ; LENSTRA JR, H. W. ; LOVÁSZ, L. Factoring polynomials with rational coefficients. **Mathematische Annalen**, Berlin, v. 261, n. 4, p. 515–534, Dec. 1982.

[37] SCHNORR, C. P. ; EUCHNER, M. Lattice basis reduction: improved practical algorithms and solving subset sum problems. **Mathematical Programming**. Berlin, 1 v. 66, n. 1-3, p. 181–199, Sept. 1994.

[38] KORKINE, A. ; ZOLOTAREFF, G. Sur les formes quadratiques. **Mathematische Annalen**, Berlin, v. 6, n. 3, p. 366–389, Sept. 1873.

[39] MINKOWSKI, H. Über die positiven quadratischen formen und über kettenbruchähnliche algorithmen. **Journal für die reine und angewandte Mathematik**, Berlin, v. 107, p. 278–297, 1891.

[40] COHEN, H. **A Course in computational algebraic number theory**. Berlin: Springer-Verlag, 1993. (Graduate Texts in Mathematics, 138).

[41] SCHNORR, C. P. Block reduced lattice bases and successive minima. **Combinatorics, Probability and Computing**, Cambridge: Cambridge University Press, v. 3, n. 4, p. 507–522, Dec. 1994.

[42] SCHNORR, C. P. A hierarchy of polynomial time lattice basis reduction algorithms. **Theoretical Computer Science**, Amsterdam, v. 53, n. 2-3, p. 201–224, 1987.

[43] KANNAN, R. Improved algorithms for integer programming and related lattice problems. In: ANNUAL ACM SYMPOSIUM ON THE THEORY OF COMPUTING, 15., 1983, Santa Barbara, CA. **Proceedings ...** New York: ACM, 1983. p.193–206.

[44] MICCIANCIO, D. Improving lattice based cryptosystems using the hermite normal form. In: SILVERMAN, J. H. (Ed.). **Cryptography and Lattices**. International Conference, CaLC 2001 Providence, RI, USA, March 29–30, 2001 Revised Papers. Berlin: Springer-Verlag, 2001. p. 126-145. (Lecture Notes in Computer Science, v. 2146).

[45] LEE, M. S. ; HAHN, S. G. Cryptanalysis of the GGH cryptosystem. **Mathematics in Computer Science**. Basel: v. 3, n. 2, p. 201–208, Apr. 2010.

[46] BERMAN, A. ; PLEMMONS, R. J. **Nonnegative matrices in the mathematical sciences**. Philadelphia: SIAM, 1994. (Classics in Applied Mathematics, Book 9).

[47] NGUYEN, P. Q. ; REGEV, O. Learning a parallelepiped: cryptanalysis of ggh and ntru signatures. In: VALDENAY, S. (Ed.). **Advances in Cryptology - EUROCRYPT 2006**. Proceedings of the 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006. Berlin: Springer-Verlag, 2006. p.271–288. (Lecture Notes in Computer Science, v.4004).

[48] MENEZES, A. J. ; OORSCHOT, P. C. Van ; VANSTONE, S. A. **Handbook of applied cryptography**. Boca Raton: CRC Press, 1996. (Discrete Mathematics and Its Applications).

[49] GOLDWASSER, S. ; MICALI, S. ; RACKOFF, C. The knowledge complexity of interactive proof systems. **SIAM Journal on Computing**, Philadelphia, v. 18, n. 1, p. 186–208, Feb. 1989.