Chaves Compactas para Esquemas Criptográficos Baseados em Códigos Corretores de Erros

Thales Areco Bandiera Paiva

Texto para Qualificação de Mestrado Apresentado
ao
Instituto de Matemática e Estatística
da
Universidade de São Paulo

Programa: Mestrado em Ciência da Computação

Orientador: Prof. Dr. Routo Terada

O autor recebe auxílio financeiro da CAPES

São Paulo, dezembro de 2016

Chaves Compactas para Esquemas Criptográficos Baseados em Códigos Corretores de Erros

Esta é a versão submetida ao exame de qualificação.

Comissão Julgadora:

Titulares:

- Prof. Dr. Routo Terada (orientador) - IME-USP
- Prof. Dr Jairo Zacarias Goncalves - IME-USP
- Prof. Dr. Marcos Antonio Simplicio Junior - EP-USP

Suplentes:

- Profa. Dra. Kelly Rosa Braghetto - IME-USP
- Profa. Dra. Denise Hideko Goya - CMCC-UFABC
- Prof. Dr. Denis Deratani Mauá - IME-USP

Sumário

Capítulo 1

Proposta

1.1 Introdução

O esquema de chave pública de McEliece [McE78] foi o primeiro baseado em códigos corretores de erros, e também o primeiro a usar um procedimento aleatório para a encriptação [MVOV96]. Tanto a encriptação quanto a decriptação são computacionalmente mais eficientes que as respectivas de outros esquemas baseados no problema do logaritmo discreto. Além disso, o esquema é baseado no problema $\mathcal{NP}$-difícil da decodificação por síndrome [BMVT78], que é conjecturado tipicamente difícil, e é teoricamente mais difícil que o problema do logaritmo discreto. Porém, os 2 fatores a seguir foram críticos para que o esquema recebesse pouca atenção na época de sua publicação.

- As chaves públicas são muito grandes. Para níveis de segurança maiores do que 100, têm tamanho maiores do que 100 quilobytes [BCS13, Tabela 1.1].
- A crença na impossibilidade de um esquema de assinatura sobre ele, justificada pelo próprio McEliece. Porém em 2001, Courtois, Finiasz, e Sendrier [CFS01] mostram como construir um tal esquema de assinatura.

Depois de anos sem atenção, o esquema de McEliece passa a ser estudado pois, ao contrário do RSA [RLR78] e das Curvas Elípticas [Mil86], esse esquema resiste a ataques quânticos baseados no algoritmo de Peter Shor [Sho97]. Com o avanços da computação quântica, os principais algoritmos de chave pública em uso ficam cada vez mais vulneráveis, o que justifica as buscas por algoritmos criptográficos resistentes a ataques quânticos. Essa busca dá origem à área chamada Criptografia Pós-Quântica, que estuda, além de esquemas baseados em códigos corretores de erros, esquemas baseados em *hashes*, reticulados, e equações quadráticas multivariadas [BBD09].

Os principais esquemas de chave pública baseados em códigos corretores de erros são os de McEliece [McE78] e de Niederreiter [Nie86]. Os esquemas têm segurança equivalente quando o adversário não tem nenhuma informação sobre a mensagem transmitida, ou quando são usadas conversões IND-CCA2 [KI01]. E, tanto por ser mais antigo, quanto por ser mais facilmente descrito, os resultados apresentados nas próximas seções são baseados no esquema de McEliece.

No esquema de McEliece, a chave secreta de Alice é um código linear $\mathcal{G}$ com capacidade de correção de t erros para o qual Alice conhece um decodificador eficiente. A chave pública de Alice, é uma matriz geradora de $\mathcal{G}$, possivelmente embaralhada de forma que seja difícil usá-la para recuperar um decodificador para $\mathcal{G}$. Para mandar uma mensagem $\boldsymbol{m}$ a Alice, codifique $\boldsymbol{m}$ usando a chave pública de Alice e adicione t erros à codificação, obtendo c. Então Alice, que é a única portadora de um decodificador eficiente para $\mathcal{G}$, decodifica c e obtém $\boldsymbol{m}$.

Uma das principais linhas de pesquisa em Criptografia com códigos corretores de erros é diminuir o tamanho das chaves do esquema de McEliece através da escolha adequada da família de códigos associada. Originalmente, McEliece sugeriu o uso de códigos de Goppa binários e irredutíveis, que, apesar de não sofrerem ataques críticos, resultam em grandes chaves. O motivo principal

por que códigos de Goppa, de taxa[1] não não muito alta [FGUO+13], ainda resistem aos ataques é que não se conhece algoritmo eficiente para distinguir entre matrizes geradoras de códigos de Goppa e de matrizes geradoras de códigos aleatórios. Algumas outras famílias de códigos foram propostas, como as dos códigos BCH quase cíclicos [Gab05], alternantes quase cíclicos [BCGO09], códigos LDPC [SMR00], e códigos de Goppa quase diádicos [MB09]. Apesar de obterem uma boa redução do comprimento da chave, todas foram mostradas inseguras depois de alguns anos [OTD10, FOPT10, FOP+16].

Em 2013, uma nova variante do Esquema de McEliece, chamada QC-MDPC McEliece, foi proposta por Misoczki et al. [MTSB13]. Esta variante promete chaves públicas de 4801 bits para um nível de segurança de 80-bits, e tem uma boa redução de segurança. A iniciativa europeia PQCRYPTO, para o desenvolvimento de criptografia pós-quântica, na revisão de 2015 de seu documento com recomendações de esquemas pós-quânticos [ABB+], considera essa variante como "sob análise", junto com a variante Stehlé-Steinfeld [SS11] do esquema NTRU [HPS98].

Até meados de 2016, o QC-MDPC McEliece não sofrera ataques críticos. Porém, na conferência Asiacrypt de 2016, Guo, Johansson, e Stankovski [GJS16] mostram um ataque de reação eficiente para a recuperação de chave de um esquema QC-MDPC. O ataque de reação contra o QC-MDPC McEliece se baseia no fato de que os decodificadores de códigos QC-MDPC são iterativos e podem falhar, e, quando isso ocorre, o destinatário pede para o remetente reenviar a mensagem com outro erro inserido. A observação dos autores é que a probabilidade de o decodificador falhar para um erro $\boldsymbol{e}$ é significativamente menor quando $\boldsymbol{e}$ e a chave secreta compartilham certas propriedades. Assim, uma atacante pode enviar um grande número de mensagens encriptadas a Alice de forma a extrair informações que serão usadas para reconstruir a sua chave privada.

A chave secreta do esquema QC-MDPC McEliece é um vetor de n posições e peso w, denotado por $\boldsymbol{h}$. Porém basta descobrir a primeira metade de $\boldsymbol{h}$, denotado por $\boldsymbol{h}_0$, e usar a chave pública para recuperar a chave privada, como pode ser visto na Seção 3. O ataque de reação recupera o conjunto de distâncias circulares entre entradas não nulas de $\boldsymbol{h}_0$, chamado seu espectro, e usa esse espectro para reconstruir a chave. Como exemplo, o espectro do vetor $[1\,1\,0\,0\,0\,1\,0]$ é o conjunto $\{1, 2, 3\}$. Note que a distância entre as posições 1 e 6 não é 5, mas sim 2, pois trata-se de distâncias circulares, que para um vetor v, sempre são menores do que $\|v\|/2$.

Nosso objetivo é analisar o ataque de reação contra o QC-MDPC McEliece sob as três perspectivas discutidas a seguir.

Reconstrução de $\boldsymbol{h}_0$ com o espectro recuperado com erros

O algoritmo de reconstrução de $\boldsymbol{h}_0$ mostrado pelos autores do ataque precisa que o espectro tenha sido recuperado sem erros. Dependendo do decodificador usado, pode ser que sejam necessários muitos pedidos de decodificação para obter a separação entre as classes de erros.

Diminuir o número de decodificações necessárias para atacar conversões CCA2

Este objetivo engloba o anterior, pois, se houver meios de reconstruir $\boldsymbol{h}_0$ mesmo com erros de recuperação do espectro, o ataque aplicado a conversões CCA2 ficaria mais eficiente.

Atacar QC-MDPC McEliece com decodificadores por decisões suaves

Os autores do ataque de reação conjecturaram que o ataque serviria para o QC-MDPC McEliece implementado com decodificação com decisões suaves. Pelos nossos testes, também conjecturamos que esses decodificadores podem ser atacados. E se esse for o caso, a variante *Soft McEliece* [BSC16] também deve ser mostrada insegura.

[1] A taxa de um código linear de dimensão k e comprimento n é igual a k/n.

1.2 O que foi feito

Nesta seção, trato em alto nível do que fizemos até agora. Os resultados e detalhes de implementação requerem certos pré-requisitos sobre o ataque de reação aos códigos QC-MDPC, portanto não caberiam aqui. Assim a Seção 4 ao final desse texto é dedicada aos resultados preliminares.

1.2.1 Seminários

Desde que começamos a estudar o esquema de McEliece, apresentei os seguintes seminários internos, de que participaram o Professor Routo, seu aluno de iniciação científica Gervásio, e nosso colega doutorando Gláucio, que é aluno do professor Paulo Barreto.

1. Códigos de Goppa binários e irredutíveis.
2. Esquema de McEliece.
3. Ataque por criptanálise algébrica aos Códigos de Goppa quase p-ádicos.
4. Conversões de segurança para o Esquema de McEliece.
5. Ataque de reação contra o QC-MDPC McEliece.

Também cursei uma disciplina de segurança da informação na Escola Politécnica, ministrada pelo Professor Marcos Simplício, em que apresentei um seminário sobre o esquema de McEliece, e um artigo sobre este trabalho. Todo o material elaborado para a matéria está disponibilizado em github.com/thalespaiva/segurinfo.

1.2.2 Código

Até agora, escrevi código em C, SageMath, e Python 3 para implementar:

1. o esquema de McEliece com códigos de Goppa em SageMath;
2. a criação, codificação e decodificação de códigos LDPC, MDPC, e QC-MDPC em C e SageMath;
3. o ataque de reação aos códigos QC-MDPC em C.

Quando possível, usei SageMath ou Python 3 pois o código é mais legível, e potencialmente mais didático. Mas como o ataque de reação é computacionalmente custoso, implementei em C, usando OpenMP para definir partes concorrentes.

1.2.3 Dissertação

Boa parte das próximas seções devem fazer parte da dissertação, embora aqui apareçam de forma resumida, com seções mescladas. A monografia está em constante evolução no repositório github.com/thalespaiva/msc. A estrutura esperada e curtas explicações de cada seção são dadas a seguir.

1 Introdução

2 Preliminares

3 Fundamentos da Teoria de Códigos

3.1 Teoria de Códigos
Introduz os conceitos básicos de Teoria de Códigos, como espaço de mensagens, códigos identificadores ou corretores de erros, codificação e decodificação, canais ruidosos.

3.2 Modelos de Canais Ruidosos
Define os dois modelos de canal ruidoso que são úteis para a criptografia: canal binário simétrico e canal de ruído branco gaussiano aditivo. O primeiro modelo é usado em praticamente todas as variantes do esquemas

de McEliece. O segundo modelo é usado apenas no esquema de McEliece com erros suaves, ou contínuos, de Baldi [BSC16].

3.3 Códigos Lineares

Define conceitos fundamentais como códigos lineares, peso, e distância de Hamming. Mostra cotas para distâncias, e capacidade de correção de códigos lineares. Também define os principais problemas como a decodificação por síndrome, e a busca por palavras de baixo peso.

4 Algumas Famílias de Códigos Notáveis

4.1 Códigos de Repetição

Essa curta seção define os códigos de repetição. Apesar de simples, esta seção não só tem caráter didático, pois ajuda o leitor a se familiarizar, com códigos lineares, mas será útil na análise dos decodificadores de códigos LDPC por passagem de mensagem. Isso pois nós de variáveis nos grafos de Tanner de códigos LDPC, ou MDPC, são vistos como decodificadores de códigos de repetição [RL09, pg. 214].

4.2 Códigos de Verificação Única de Paridade

A justificativa é similar à da seção anterior, com a diferença de que códigos de verificação única de paridade estão relacionados aos nós de verificação em grafos de Tanner de códigos LDPC.

4.3 Códigos Quase Cíclicos

Introduz os códigos quase cíclicos, que são a principal forma usada para diminuir as chaves públicas nos esquemas de McEliece, como em códigos QC-MDPC.

4.4 Códigos de Goppa

Introduz os códigos algébricos de Goppa usados no esquema original de McEliece, mostrando sua capacidade de correção, o algoritmo de decodificação de Patterson [Pat75], e mostrando como gerar instâncias aleatórias.

4.5 Códigos LDPC

Define os códigos LDPC de Gallagher [Gal62], mostrando como gerar instâncias aleatórias, mostrando decodificadores por passagem de mensagem, tanto por decisão abrupta quanto por decodificação suave.

5 Esquemas de McEliece e Niederreiter

5.1 Esquema de McEliece

Apresenta detalhadamente o esquema original de McEliece. Discute os tamanhos das chaves.

5.2 Esquema de Niederreiter

Similar à seção anterior para o esquema de Niederreiter.

5.3 Segurança dos Esquemas

5.3.1 Equivalência entre dois Esquemas

Mostra como o esquema de McEliece tem segurança equivalente à do esquema de Niederreiter, que pode ser visto como o seu dual.

5.3.2 Redução de Segurança

Discute as hipóteses em que é baseada a segurança do esquema de McEliece e mostra a redução do problema da decodificação ao problema de quebrar um texto cifrado no esquema de McEliece.

5.3.3 Possíveis Ataques ao Esquema de McEliece Original

Mostra os ataques não-críticos e críticos. Os não críticos são por decodificação por conjunto de informação, e busca por palavras de baixo peso. Os críticos são ataques de decodificação com informação parcial sobre a mensagem, ou ataques de reação [S$^+$00, KI01]. Mostrando que o e

5.3.4 Conversões de Segurança

Discute por que o esquema de McEliece não pode ser usado, a menos que para troca de chaves, sem uma conversão de segurança. E mostra ao menos duas conversões, como as α e γ de Kobara e Imai [KI01], que transformam o esquema de McEliece num esquema IND-CCA2. Sugiro mostrar a α por simplicidade, e a γ pois é a mais eficiente. Ambas são mostradas seguras sob o modelo de Oráculos Aleatórios.

6 Algumas Variantes com Chaves Compactas Quebradas Mostra variantes como as que usam códigos de Goppa Quase Cíclicos, códigos de Goppa Quase p-ádicos, e códigos LDPC. A ideia é dar ao leitor certa intuição sobre os tipos de vulnerabilidade que esquemas de McEliece com chaves compactas podem ter.

7 Esquema de McEliece com códigos QC-MDPC

7.1 Códigos QC-MDPC

Introduz os códigos QC-MDPC. Mostra algoritmos para a construção, codificação, e decodificação usando esses códigos.

7.2 QC-MDPC McEliece

Mostra como usar os códigos QC-MDPC no esquema de McEliece, e compara com o esquema de McEliece original. Mostra os tamanhos de chave e discute a sua redução de segurança.

7.3 Ataque de Reação aos Códigos QC-MDPC

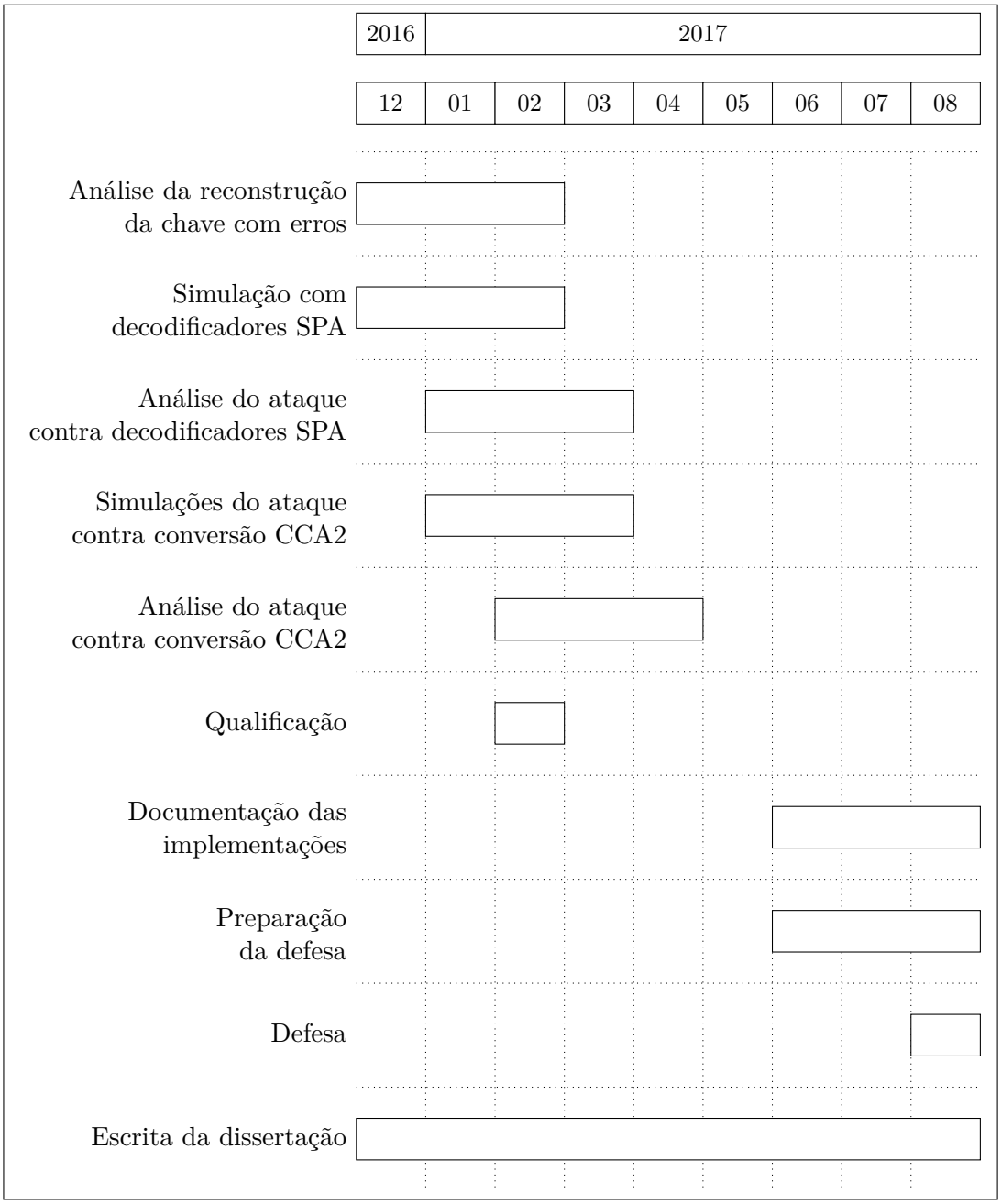
Introduz o ataque de reação ao QC-MDPC McEliece. Discute o desempenho do ataque e possíveis melhorias.

8 Melhorias sobre o Ataque de Reação

Mostra nossas propostas de modificação do algoritmo. Faz tanto a análise teórica quanto a prática dos algoritmos modificados.

9 Conclusão

1.3 O que será feito



1.4 Alguns resultados elementares de Teoria de Códigos

O esquema de McEliece é baseado em códigos corretores de erros, que são meios eficientes de transmitir mensagens, de uma fonte a um receptor, através de um canal, possivelmente ruidoso. Quando um canal de transmissão não é ruidoso, a meta é codificar uma mensagem de modo que a mensagem transmitida tenha o mínimo possível de símbolos, e a mensagem original possa ser decodificada da transmitida. Por sua vez, numa transmissão feita por um canal ruidoso, a meta é

codificar a mensagem de modo que a mensagem original possa ser recuperada, mesmo que alguns símbolos sejam alterados por conta de ruídos intrínsecos ao canal.

A Figura 1.4.1 exemplifica a transmissão de uma mensagem $\boldsymbol{m}$ por um canal ruidoso. Na figura, $\mathcal{M}$ é o conjunto de mensagens possíveis, E é o algoritmo de codificação, $\boldsymbol{e}$ é o erro aleatório do canal adicionado bit a bit a $\boldsymbol{c}$, e D é o algoritmo de decodificação. Naturalmente, espera-se que E e D rodem em tempo polinomial, e E é tipicamente determinístico, enquanto D pode ser probabilístico. Idealmente, queremos que $\boldsymbol{m} = \hat{m}$ ao final da transmissão. Porém, se o erro $\boldsymbol{e}$ fizer $\boldsymbol{c'}$ ser muito diferente de $\boldsymbol{c}$, pode ser que o algoritmo D não seja capaz de recuperar a mensagem $\boldsymbol{m}$ corretamente.

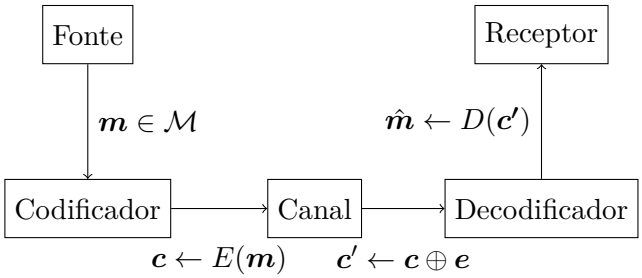


Figura 1.4.1: *Transmissão de uma palavra $\boldsymbol{m}$ através de um canal ruidoso.*

Definição 1.4.1. Um *código binário* $[n, k]$*-linear* é um subespaço vetorial de dimensão k do espaço $\mathbb{F}_2^n$.

Por serem subespaços vetoriais, códigos lineares têm representações compactas através da imagem de matrizes, ou equivalentemente, através do núcleo de matrizes. Destas representações, seguem naturalmente as seguintes definições.

Definição 1.4.2. Seja $\mathcal{C}$ um código binário $[n, k]$-linear. Se $\mathcal{C}$ for igual ao espaço gerado por combinações lineares das linhas de uma matriz G de $\mathbb{F}_2^{k\times n}$, dizemos que G é uma *matriz geradora* de $\mathcal{C}$.

Analogamente, se $\mathcal{C}$ é o núcleo de uma matriz H de $\mathbb{F}_2^{n-k\times n}$, dizemos que H é uma *matriz de paridade* de $\mathcal{C}$.

Assim, se G e H são matrizes geradoras quaisquer de um código $[n, k]$-linear $\mathcal{C}$, temos que:

$$\mathcal{C} = \{\boldsymbol{m}G : \boldsymbol{m} \in \mathbb{F}_2^k\} = \{\boldsymbol{c} \in \mathbb{F}_2^n : \boldsymbol{c}H^T = 0\}.$$

Uma matriz de paridade tem esse nome pois, para um vetor $\boldsymbol{c} = [c_1 \dots c_n]$ pertencer a um código que admite matriz de paridade H, as seguintes equações devem ser satisfeitas:

$$\begin{array}{ccccccccc} c_1h_{11} & \oplus & c_2h_{12} & \oplus & \dots & c_nh_{1n} & \oplus & = & 0 \\ c_1h_{21} & \oplus & c_2h_{22} & \oplus & \dots & c_nh_{2n} & \oplus & = & 0 \\ \vdots & & \vdots & & \dots & & \oplus & = & \vdots \\ c_1h_{r1} & \oplus & c_2h_{r1} & \oplus & \dots & c_nh_{rn} & \oplus & = & 0 \end{array}$$

que por serem equações sobre $\mathbb{F}_2$, são ditas equações de paridade.

Para verificar se um dado vetor pertence a um código, podemos multiplicar à esquerda por H^T e ver se chegamos ao vetor 0 ou não. Mas para a decodificação, muitas vezes é útil saber quais as equações de paridade que não são satisfeitas por um dado vetor, que é o conceito de síndrome.

Definição 1.4.3. A *síndrome* de um vetor $\boldsymbol{v}$, denotada por $s(\boldsymbol{v})$, em relação a um código que admite matriz de paridade H, é

$$s(\boldsymbol{v}) = \boldsymbol{v}H^T.$$

Claro que a síndrome de um vetor é 0 se e somente se este vetor pertence ao código.

Dois conceitos importantes para o estudo de códigos são o de peso e distância. Com eles, podemos tratar de $\mathbb{F}_2^n$ como um espaço métrico.

Definição 1.4.4. O *peso de Hamming* de um vetor $\boldsymbol{v}$, denotado por $\mathrm{w}(\boldsymbol{v})$, é o número total de suas coordenadas não nulas.

Definição 1.4.5. A *distância de Hamming* entre dois vetores $\boldsymbol{u}$ e $\boldsymbol{v}$, denotada por $\mathrm{dist}(\boldsymbol{u}, \boldsymbol{v})$, é o número total de coordenadas em que $\boldsymbol{u}$ e $\boldsymbol{v}$ diferem. Equivalentemente $\mathrm{dist}(\boldsymbol{u}, \boldsymbol{v}) = \mathrm{w}(\boldsymbol{u} \oplus \boldsymbol{v})$.

Agora estamos aptos a definir um problema fundamental em teoria de códigos, e o problema $\mathcal{NP}$-difícil[BMVT78] em que é baseado o esquema de McEliece.

Definição 1.4.6. O *problema da decodificação por distância mínima* é: dado um código $[n, k]$-linear $\mathcal{C}$, e um vetor c' de $\mathbb{F}_2^n$, encontre o vetor c de $\mathcal{C}$ tal que $\mathrm{dist}(c, c')$ é mínima.

O esquema de McEliece original usa códigos de Goppa. Estes são códigos algébricos que são bons candidatos ao uso criptográfico, mas não são muito interessantes para chaves compactas. Algumas de suas propriedades são listadas a seguir.

- Cada código é gerado por um polinômio irredutível g de $\mathbb{F}_{2^m}[x]$.
- A capacidade de correção é igual ou superior ao grau t de g.
- É fácil construir decodificadores de t erros eficientes conhecendo g.
- Não se conhece algoritmo eficiente para construir decodificadores sem conhecer g.
- É difícil em geral distinguir matrizes geradoras de Goppa e geradoras de aleatórios, embora seja possível para k próximo de n [FGUO+13].

Capítulo 2

Esquema de McEliece

Definição 2.0.1 (Esquema de McEliece)**.** Um *Esquema de McEliece* é uma tripla de algoritmos (KeyGen, Enc, Dec), em que cada algoritmo é definido como os seguintes.

KeyGen é o algoritmo para geração de um par de chaves. Dado um nível de segurança λ, siga os seguintes passos.

1. Escolha uma tripla de parâmetros (n, t, k) de uma família $\mathcal{F}$ de códigos de Goppa que suporta um nível de segurança λ usando, por exemplo, a Tabela 2.0.1.
2. Tome $\overline{G}$ e g, respectivamente, uma matriz geradora, e um polinômio de Goppa, de um mesmo código $\mathcal{G}$ escolhido aleatoriamente da família $\mathcal{F}$.
3. Escolha aleatoriamente S, uma matriz $k \times k$ não singular, e P, uma matriz de permutação $n \times n$. Ambas têm entradas em $\mathbb{F}_2$.
4. Faça $G \leftarrow S\overline{G}P$, e $\hat{G} \leftarrow S\overline{G}$.
5. Devolva o par de chaves secreta e pública, dados respectivamente por

$$K_{\text{Sec}} \leftarrow (\hat{G}, P, g), \quad \text{e} \quad K_{\text{Pub}} \leftarrow (G, t).$$

Enc é o algoritmo de encriptação. Dada uma mensagem $\boldsymbol{m}$, e a chave pública do destinatário K_{Pub}, siga os seguintes passos.

1. Separe os elementos da chave pública em $(G, t) \leftarrow K_{\text{Pub}}$.
2. Tome $\boldsymbol{e} \leftarrow$ vetor aleatório de $\mathbb{F}_2^n$ de peso t
3. Devolva o vetor $\boldsymbol{c} \leftarrow mG \oplus \boldsymbol{e}$

Dec é o algoritmo de decriptação. Dado um texto cifrado $\boldsymbol{c}$ e a chave secreta do destinatário K_{Sec}, siga os seguintes passos.

1. Separe os elementos da chave secreta em $(\hat{G}, P, g) \leftarrow K_{\text{Sec}}$.
2. Obtenha o decodificador Φ a partir de g.
3. Calcule $\boldsymbol{c}_1 \leftarrow \boldsymbol{c}P^{-1}$. Assim, $\boldsymbol{c}_1$ representa

$$\boldsymbol{c}P^{-1} = (\boldsymbol{m}G \oplus \boldsymbol{e})P^{-1} = \boldsymbol{m}\hat{G} \oplus \boldsymbol{e}P^{-1}.$$

4. Calcule $\boldsymbol{c}_2 \leftarrow \Phi(\boldsymbol{c}_1)$ que é igual a $\boldsymbol{m}\hat{G}$.
5. Devolva $\boldsymbol{m}$ que é a solução do sistema sobredeterminado $\boldsymbol{m}\hat{G} = \boldsymbol{c}_2$.

A Tabela 2.0.1, adaptada do artigo de Bernstein, Chou, e Schwabe [BCS13], mostra parâmetros de códigos de Goppa para alguns níveis de segurança.

λ	n	k	t	m	Tamanho da chave pública (Bytes)
81	2048	1751	27	11	65006
95	2048	1608	40	11	88440
105	2480	1940	45	12	130950
119	2690	2018	56	12	169512
146	3408	2604	67	12	261702
187	4624	3389	95	13	523177
263	6960	5413	119	13	1046739

Tabela 2.0.1: *Parâmetros de famílias de códigos de Goppa para cada nível de segurança λ [BCS13].*

O esquema de McEliece se baseia em duas hipóteses:

1. Não existe algoritmo eficiente capaz de decodificar códigos lineares aleatórios.
2. Para a família de códigos $\mathcal{F}_t^{n,k}$ escolhida, as matrizes públicas geradas são indistinguíveis de matrizes aleatórias.

A primeira hipótese é relativamente bem aceita, pois sabe-se que o problema de decodificar códigos lineares aleatórios é $\mathcal{NP}$-difícil. Porém, a única família de códigos que ainda resiste indistinguível é a dos códigos de Goppa.

2.1 Segurança

O principal ataque que o esquema de McEliece sofre é chamado de decodificação por conjunto de informação (ISD) - *Information Set Decoding*. Seja I um conjunto de k índices quaisquer do vetor c. Se para todo índice i de I, temos $e_i = 0$, e se a submatriz G_I de G formada pelas colunas k de índices em i for inversível, um atacante pode obter facilmente a mensagem encriptada m. Basta fazer a multiplicação $m = c_I G_I^{-1}$, em que c_I denota o vetor formado pelas colunas de c com índices em I. Por sorte este ataque não é factível na prática, já tem complexidade limitada inferiormente por $\binom{n}{k}/\binom{n-t}{k}$.

Vejamos como a captura de certas informações pode deixar o ataque por ISD mais eficiente. As próximas seções são adaptações do trabalho de Kobara e Imai [KI01].

2.1.1 Ataque por mensagem parcialmente conhecida

Considere o cenário em que Beto envia $c = mG \oplus e$ à Alice, e suponha que a adversária Eva conhece os k_r últimos bits de m. Sejam de m_l e m_r os k_l primeiros e os k_r últimos bits de m, respectivamente. Então

$$c = m_l G_l \oplus m_r G_r \oplus e,$$

onde G_l é a matriz formada pelas primeiras linhas de G, e G_r a matriz formada pelas r últimas.

Como Eva conhece $m_r G_r$, ela pode calcular $c' = c \oplus m_r G_r$. E assim, a dificuldade de encontrar m_l com a equação

$$c' = m_l G_l \oplus e,$$

é a mesma de recuperar uma mensagem encriptada usando parâmetros (n, k_l, t).

Denotamos por $WF_{\mathrm{KPP}}(n, k, t, k_l)$ o esforço necessário para Eva recuperar uma mensagem m num esquema de McEliece com parâmetros (n, k, t), quando ela desconhece apenas k_l bits da m. A abreviação KPP vem de *known partial plaintext*, nome comumente usado para este tipo de ataque.

E assim, uma cota superior natural para $\mathbf{WF}_{\mathrm{KPP}}$ é

$$\mathbf{WF}_{\mathrm{KPP}}(n,k,t,k_l) \leq \mathbf{WF}_{\mathrm{ISD}}(n,k_l,t) \leq k_l^3 \frac{\binom{n}{k_l}}{\binom{n-t}{k_l}}.$$

2.1.2 Ataque por mensagem relacionada

Para este ataque, abreviado como RMA, de *related message attack*, considere que Beto envia a Alice duas mensagens m_1 e m_2, cujos textos cifrados são c_1 e c_2, respectivamente. Suponha que Eva conhece um vetor δ_m que relaciona as mensagens enviadas, como por exemplo $\delta_m = m_1 \oplus m_2$. Então, a soma dos cifrados é

$$\begin{aligned} c_1 \oplus c_2 &= (m_1 G \oplus e_1) \oplus (m_2 G \oplus e_2) \\ &= (m_1 \oplus m_2)\, G \oplus (e_1 \oplus e_2) \\ &= \delta_m G \oplus (e_1 \oplus e_2). \end{aligned}$$

E Eva pode calcular

$$\overline{e} = c_1 \oplus c_2 \oplus \delta_m G,$$

que é o vetor resultante da soma $e_1 \oplus e_2$.

A importância do vetor $\overline{e}$ na recuperação das mensagnes, é que, se $\overline{e}_i = 0$, a probabilidade de as i-ésimas entradas dos erros e_1 e e_2 também serem 0 é

$$\begin{aligned} \Pr(e_{1i} = 0 \wedge e_{2i} = 0 \mid \overline{e}_i = 0) &= \frac{\Pr(e_{1i} = 0 \wedge e_{2i} = 0 \wedge \overline{e}_i = 0)}{\Pr(\overline{e}_i = 0)} \\ &= \frac{(n-t)^2}{(n-t)^2 + t^2}. \end{aligned}$$

Essa probabilidade em geral é bem alta. Tome, por exemplo, $n = 6960$ e $t = 119$, que são parâmetros de uma família de códigos de Goppa que permitem um nível segurança de 263. A probabilidade será

$$\Pr(e_{1i} = 0 \wedge e_{2i} = 0 \mid \overline{e}_i = 0) \geq 99.76\%.$$

Com isso, Eva pode usar o ISD escolhendo aleatoriamente k coordenadas que são nulas em $\overline{e}$.

2.1.3 Ataque de reação

Neste ataque, a adversária Eva tenta obter alguma informação confidencial através da reação de Alice sobre um pedido de decriptação. Considere que um remetente envia um texto cifrado $\overline{c}$, não necessariamente válido. Alice pode:

- aceitar $\overline{c}$, após decriptá-lo e obter uma mensagem válida;
- rejeitar $\overline{c}$, por não conseguir decriptá-lo, ou por não ser o cifrado de uma mensagem válida.

Suponha que Eva pode enviar textos a Alice, e observar a sua reação a eles. Isso ocorre, por exemplo, se Alice, ao não conseguir decriptar um texto, pede ao remetente que encripte a mensagem novamente e envie o novo cifrado.

Para decifrar o cifrado $c = mG \oplus e$, Eva pode usar as reações de Alice para reconstruir e, e

recuperar m usando o Algoritmo 1 a seguir.

Algoritmo 1: Ataque de reação ao esquema de McEliece.

Entrada: c texto cifrado que Eva quer decifrar.
K_{PUB} a chave pública de Alice.

Saída: m a decifração de c.

início
  $e \leftarrow [0, \ldots, 0]$
  para *cada* $i = 1, \ldots, n$ **faça**
    $\overline{c} \leftarrow c$
    $\overline{c}_i \leftarrow c_i \oplus 1$
    se *Alice aceita* $\overline{c}$ **então**
      $e_i = 1$
  $(G, t) \leftarrow K_{\text{PUB}}$
  $m \leftarrow$ solução do sistema sobredeterminado $mG = c \oplus e$
  devolva m

Note que o ataque descrito pelo Algoritmo 1 pode ser evitado se Alice rejeitar qualquer cifrado que não tenha sido criado com um erro de peso t. Mas mesmo nesse caso, Eva pode gerar $\overline{c}$ invertendo simultaneamente dois bits de c a cada iteração. E, quando Alice aceita $\overline{c}$, Eva sabe que exatamente um dos bits alterados está com erro.

2.2 Conversões de segurança

Apesar de a encriptação não ser determinística, é fácil ver que o esquema de McEliece não provê indisguibilidade de textos legíveis escolhidos (IND-CPA), que é a noção de segurança mais elementar para esquemas de chave pública. Para ver isso, tome duas mensagens $\boldsymbol{m}_1$ e $\boldsymbol{m}_2$, e um cifrado $\boldsymbol{c}$ de uma das mensagens, escolhida ao acaso. Para determinar qual mensagem foi encriptada para obter $\boldsymbol{c}$, basta calcular $\delta_1 = \boldsymbol{m}_1 G \oplus \boldsymbol{c}$, e $\delta_2 = \boldsymbol{m}_2 G \oplus \boldsymbol{c}$, e se $\mathrm{w}(\delta_1) = t$, então $\boldsymbol{c} = \boldsymbol{m}_1 G$. Caso contrário, necessariamente $\delta_2 = t$, e $\boldsymbol{c} = \boldsymbol{m}_2 G$.

Assim, para usar o esquema de McEliece de forma segura, é necessário fazer conversões de segurança. O ideal é atingir IND-CCA2, pois assim nenhum dos ataques críticos apresentados nas seções passadas são possíveis.

Há conversões genéricas, como a de Pointcheval [Poi00], e a de Fujisaki e Okamoto [FO99], que convertem esquemas PTOWF, e OWE, respectivamente, em esquemas IND-CCA2. Ambas são aplicáveis ao esquema de McEliece, que é um PTOWF, e portanto um OWE. Porém, por serem genéricas, não usam o esquema de McEliece do modo mais eficiente possível, e portanto há redundância de dados maior do que necessária.

Kobara e Imai [KI01] mostram três conversões eficientes, chamadas α, β, e γ, que convertem o esquema de McEliece num esquema IND-CCA2, sob o modelo de oráculos aleatórios. As suas conversões específicas chegam a ter redundância de dados[1] até 4 vezes menor do que as genéricas, e ainda, para alguns parâmetros, a conversão γ pode ser até mais eficiente em uso de dados do que o McEliece original sem conversões, pois maximiza a quantidade de informação no vetor de erro. A Tabela 2.2.1 mostra a redundância de dados para cada conversão, com diferentes parâmetros (n, k, t) do esquema de McEliece.

[1] A redundância de dados em uma conversão é a diferença entre os comprimentos do texto cifrado e do texto legível.

Conversão	Redundância de dados $\|c\| - \|m\|$ para parâmetros (n, k, t)		
	(1024, 644, 38)	(2048, 1289, 69)	(4096, 2560, 128)
Pointcheval	1184	2208	4256
Fujisaki e Okamoto	1024	2048	4096
Kobara e Imai α e β	540	919	1696
Kobara e Imai γ	470	648	1040
McEliece original	380	759	1536

Tabela 2.2.1: *Comparação da redundância de dados para as conversões genéricas e específicas [KI01, Tabela 1 adaptada].*

Por ser a conversão mais simples, mostramos aqui a conversão α, cuja encriptação é dada pelo Algoritmo 2, e a decriptação no Algoritmo 3. Como outras conversões de segurança sob o modelo de oráculos aleatórios, a ideia central é reduzir a quebra da encriptação de mensagens aleatórias com o esquema de McEliece, que conjecturamos ser segura, à quebra da encriptação de mensagens não aleatórias, que um atacante pode conhecer parcialmente.

Algoritmo 2: Conversão IND-CCA2 de Enc [KI01]

Entrada: m mensagem que Beto quer enviar a Alice.
K_{PUB} a chave pública de Alice.

Saída: c o texto cifrado da mensagem m.

início
- $(G, t) \leftarrow K_{\text{PUB}}$
- $r \leftarrow$ número aleatório ≈ 160 bits
- $z \leftarrow \text{Hash}(r \| m)$
- $y \leftarrow \text{PRNG}(z) \oplus (r \| m)$
- $e_z \leftarrow \text{IntegerToErrorVector}(z, t)$
- $c \leftarrow (yG \oplus e_z)$
- **devolva** c

Algoritmo 3: Conversão IND-CCA2 de Dec [KI01]

Entrada: c texto cifrado recebido por Alice.
K_{SEC} a chave secreta de Alice.

Saída: m a decifração de c.

início
- $y \leftarrow \text{Dec}(\text{Esquerda}(c, k), K_{\text{SEC}})$
- $e_z \leftarrow \text{Esquerda}(\text{c}, \text{k}) \oplus yG$
- $z \leftarrow \text{ErrorVectorToInteger}(e_z)$
- $r \| m \leftarrow \text{PRNG}(z) \oplus (y)$
- **se** $z = \text{Hash}(r \| m)$ **então**
  - **devolva** m
- **senão**
  - **devolva** $\perp$

Capítulo 3

Esquema de McEliece com códigos QC-MDPC

Em 2013, uma nova variante do Esquema de McEliece foi proposta por Misoczki et al. [MTSB13]. Esta variante usa códigos QC-MDPC, que são códigos lineares binários que admitem uma matriz de paridade quase cíclica e moderadamente esparsa, em contraste com as de baixa densidade dos códigos LDPC de Gallager [Gal62].

A ideia dos autores foi fortalecer o esquema de McEliece com códigos LDPC [SMR00], aumentando a densidade da matriz de paridade de forma que a busca por linhas esparsas da matriz de paridade ficasse intratável, obtendo códigos chamados MDPC. Mas como as chaves públicas usando códigos MDPC ainda eram muito grandes, os autores mostraram como introduzir certa ciclicidade nas matrizes, de modo a manter o esquema seguro, e obter chaves compactas comparáveis às do RSA.

Além da variante de Misoczki et al. há duas outras variantes que usam códigos QC-MDPC, como a chamada QC-MDPC suave [BSC16], e a QC-MDPC p-ária [GJ16]. Ainda não se sabe como o ataque de Guo e Johansson afeta a segurança desses dois esquemas últimos esquemas.

3.1 Códigos QC-MDPC

Definição 3.1.1 (Códigos MDPC e QC-MDPC [MTSB13])**.** Um *código* (n, r, w)*-MDPC* é um código linear de comprimento n, e codimensão [1] r, que admite uma matriz de verificação de paridade H cujas linhas têm mesmos pesos iguais a $w = \mathcal{O}\left(\sqrt{n \log n}\right)$.

Quando H é quase cíclica, um tal código é dito um *código* (n, r, w)*-QC-MDPC*. Quando não há dúvidas de que código estamos tratando, denotamos por n_0 o número de blocos cíclicos de H. E como blocos cíclicos são matrizes quadradas $r \times r$, é claro que $n_0 = \frac{n}{r}$.

Assim, por definição, a matriz de paridade de um código (n, r, w)-QC-MDPC é da forma

$$H = [\, H_0 \,|\, H_1 \,|\, \ldots \,|\, H_{n_0-1} \,].$$

Onde cada H_i é cíclica. Então H é completamente descrita pela sua primeira linha, e seu número de linhas r, ou equivalentemente o número de blocos n_0.

Supondo que H_{n_0-1} é inversível, o que será garantido pelo algoritmo de construção de códigos MDPC da Seção 3.1.1, é fácil verificar que uma possível matriz geradora do código de matriz de

[1] A codimensão de um código é a dimensão de seu espaço dual, ou equivalentemente o número de linhas de uma sua matriz de paridade.

paridade H é

$$G = \left[\begin{array}{c|c} & \left(H_{n_0-1}^{-1} \cdot H_0\right)^T \\ & \left(H_{n_0-1}^{-1} \cdot H_1\right)^T \\ I & \vdots \\ & \left(H_{n_0-1}^{-1} \cdot H_{n_0-2}\right)^T \end{array}\right]. \tag{3.1.1}$$

Em todo o resto deste texto, supomos que a matriz geradora G está na forma sistemática, pois não há perda de segurança quando se usa conversões IND-CCA2, como a de Kobara e Imai.

3.1.1 Construção do código

Para construir um código (n, r, w)-QC-MDPC aleatório, basta escolher aleatoriamente um vetor linha h, de comprimento n e peso w, cujas últimas r colunas gerem uma matriz quase cíclica inversível. Assim, usa-se esse vetor h como gerador da matriz quase cíclica H, de verificação de paridade. E claro que a matriz geradora poderá ser derivada de H a partir da Equação 3.1.1.

Um modo simples de agilizar a construção de códigos QC-MDPC, é observar que a álgebra das matrizes cíclicas $r \times r$ é isomorfa à dos polinômios em $\mathbb{F}_2[x]$ módulo $x^r - 1$. Dessa forma, as multiplicações e inversões de matrizes são computacionalmente eficientes.

Tipicamente, as matrizes de paridade dos códigos usados no QC-MDPC McEliece têm apenas dois blocos. Como exemplo, considere as matrizes de paridade e geradora de um código $(40, 20, 6)$-QC-MDPC mostradas, respectivamente, à esquerda e à direita na Figura 3.1.1.

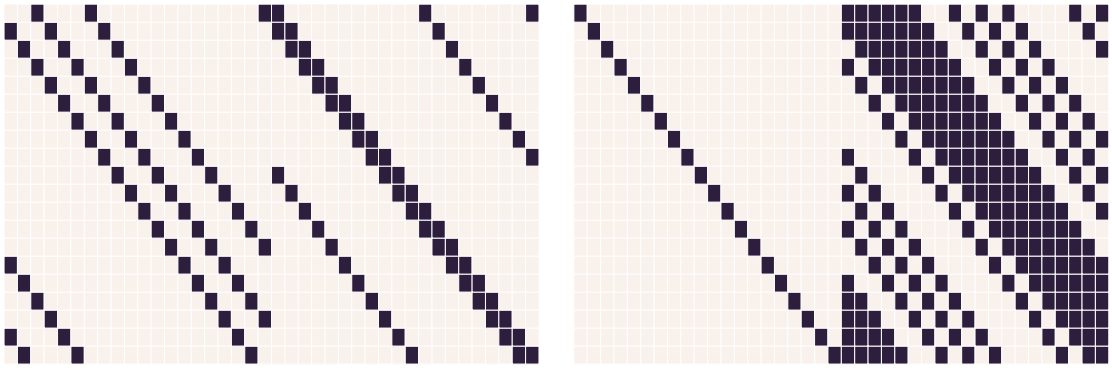


Figura 3.1.1: *Matrizes de paridade e geradora, respectivamente à esquerda e à direita, de um código* $(40, 20, 6)$*-QC-MDPC.*

3.1.2 Codificação

A codificação de um vetor m é simplesmente o vetor c, produto da multiplicação $c = mG$. E, como G tipicamente está na forma sistemática $G = [\, I \mid P \,]$, a codificação é ligeiramente simplificada para $c = m \| mP$.

3.1.3 Decodificação

Por serem baseados nos códigos LDPC, é comum decodificar códigos MDPC com algoritmos daqueles códigos. Mas como as matrizes de verificação de paridade de códigos MDPC são mais densas, é de se esperar que os algoritmos de decodificação para códigos LDPC não sejam tão eficientes. Isso motivou algumas melhorias para diminuir tanto para diminuir o número de iterações para a decodificação, quanto para diminuir a probabilidade de falha.

Há duas classes de códigos LDPC, os por decisões abruptas, e os por decisões suaves. Com algoritmos de decisão dura, a decodificação de uma mensagem transmitida é feita através de su-

cessivas inversões de seus bits, até possivelmente chegar na mensagem original. Por sua vez, os de decisão suave usam algoritmos de propagação de crença sobre a probabilidade de determinado bit da mensagem transmitida ser 1 ou não.

Para adquirir certa intuição sobre as duas classes de decodificadores, considere as Figuras 3.1.2 e 3.1.3. Elas mostram a decodificação de uma palavra $c' = c \oplus e$ para a palavra c. A primeira mostra a decodificação usando o algoritmo de *bit-flipping*, e a segunda mostra a mesma decodificação usando o algoritmo de propagação de crença. Os dois exemplos são decodificações para um código LDPC real gerado aleatoriamente. Este código tem tamanho $n = 50$, dimensão $k = 25$, e tem colunas de mesmo peso $w_c = 4$. Não é feito nenhum controle sobre o peso das linhas, nem sobre a maximização da cintura. E os algoritmos que implementei são os mesmos de Gallager.

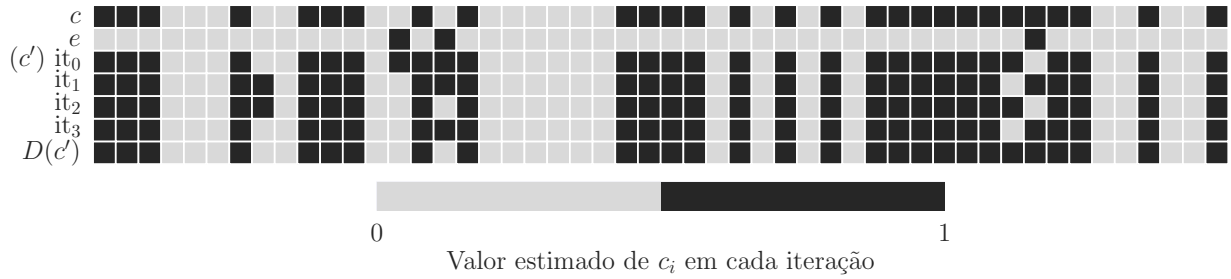


Figura 3.1.2: *Decodificação por decisão dura.*

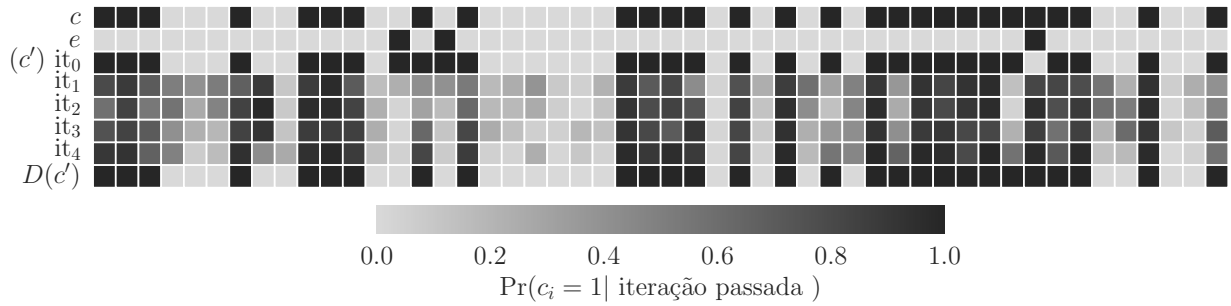


Figura 3.1.3: *Decodificação por decisão suave.*

Apesar de serem capazes de corrigir mais erros, os decodificadores por decisões suaves são significativamente mais lentos do que os por decisão abruptda. Por isso, e por terem implementação e análise mais fáceis, os decodificadores por decisão abrupta são mais comuns. Em particular, Misoczki et al. sugerem o uso de um decodificador por decisão abrupta, que é uma variação do algoritmo de *bit-flipping*, e é descrito pelo Algoritmo 15. Para uma análise experimental detalhada de variantes do algoritmo de *bit-flipping* aplicados a códigos QC-MDPC, veja o estudo de Hayse,

Maurich, e Güneysu [HVMG13].

Algoritmo 4: Variante do algoritmo de *bit-flipping* sugerido Misockzi et al. [MTSB13].

Entrada: H matriz de paridade $r \times n$ de um código LDPC.
y uma palavra transmitida a ser decodificada.
`max_it` o número máximo de iterações.

Saída: Uma decodificação estimada de y, ou $\perp$ se o número máximo de iterações for excedido.

início
  $s \leftarrow Hy$
  $\texttt{it} \leftarrow 0$
  enquanto $s \neq 0$ *e* $\texttt{it} < \texttt{max\_it}$ **faça**
    para *cada* $j = 1, 2, \ldots, n$ **faça**
      $f_j \leftarrow |\{\, i : s_i h_{ij} = 1 \text{ para } i = 1, \ldots, r \,\}|$
    $\texttt{max\_upc} \leftarrow \max\{f_j\}$
    para *cada* $j = 1, 2, \ldots, n$ **faça**
      se $f_j \geq \texttt{max\_upc} - \delta$ **então**
        $y_j \leftarrow \overline{y_j}$
    $s \leftarrow Hy$
  se $s = 0$ **então**
    devolva y
  senão
    devolva $\perp$

3.2 QC-MDPC McEliece

Definição 3.2.1 (Esquema de McEliece)**.** Um *Esquema QC-MDPC McEliece* é uma tripla de algoritmos (KeyGen, Enc, Dec), em que cada algoritmo é definido como os seguintes.

KeyGen é o algoritmo para geração de um par de chaves. Dado um nível de segurança λ, siga os seguintes passos.

1. Escolha uma tripla de parâmetros (n, r, w, t) de uma família $\mathcal{F}$ de códigos de QC-MDPC que suporta um nível de segurança λ usando, por exemplo, a Tabela 3.2.1.
2. Tome $\overline{H}$ uma matriz de paridade escolhido aleatoriamente da família $\mathcal{F}$.
3. Calcule G sistemática tal que $GH^T = 0$ usando a Equação 3.1.1.
4. Construa h_i, e g_j, as primeiras linhas de cada bloco cíclico de H de G, respectivamente. E faça
$$h \leftarrow h_0 \| h_1 \| \ldots \| h_{n_0}, \qquad e \qquad g \leftarrow g_0 \| g_1 \| \ldots \| g_{n_0 - 1}.$$
5. Devolva o par de chaves secreta e pública, dados respectivamente por
$$K_{\text{SEC}} \leftarrow (h, r), \quad \text{e} \quad K_{\text{PUB}} \leftarrow (g, t).$$

Enc é o algoritmo de encriptação. Dada uma mensagem $\boldsymbol{m}$, e a chave pública do destinatário K_{PUB}, siga os seguintes passos.

1. Separe os elementos da chave pública em $(g, t) \leftarrow K_{\text{PUB}}$.

2. Construa a matriz geradora sistemática G usando as primeira linhas de cada bloco cíclico g.
3. Tome $\boldsymbol{e} \leftarrow$ vetor aleatório de $\mathbb{F}_2^n$ de peso t
4. Devolva o vetor $\boldsymbol{c} \leftarrow m\boldsymbol{G} \oplus \boldsymbol{e}$

Dec é o algoritmo de decriptação. Dado um texto cifrado $\boldsymbol{c}$ e a chave secreta do destinatário K_{SEC}, siga os seguintes passos.

1. Separe os elementos da chave pública em $(h, r) \leftarrow K_{\text{PUB}}$.
2. Construa a matriz de paridade H usando a primeira linha h e o tamanho de cada bloco r.
3. Use o algoritmo de *bit-flipping* com a matriz esparsa H para decodificar $\boldsymbol{c}$ e obter m.
4. Se o algoritmo falhar, peça para o remetente reenviar a mensagem, que com alta probabilidade será encriptada com um padrão de erros que o decodificador pode corrigir.

Os parâmetros sugeridos para alguns níveis de segurança são descritos na Tabela 3.2.1.

Segurança	n_0	n	r	w	t	Tamanho da Chave
80	2	9602	4801	90	84	4801
128	2	19714	9857	142	134	9857
256	2	65542	32771	274	264	32771

Tabela 3.2.1: *Parâmetros sugeridos para cada nível de segurança.*

Note que, para códigos QC-MDPC, não há necessidade das matrizes embaralhadoras S e P. Isso pois, com conversões IND-CCA2, não há vazamentos sobre informações da mensagem mesmo usando G sistemática, e é supostamente difícil de se obter H a partir de G.

A redução de segurança do QC-MDPC McEliece se baseia na hipótese de que é difícil distinguir códigos QC-MDPC de códigos apenas quase cíclicos, que é um problema próximo ao de encontrar palavras de pesos baixos em códigos lineares. E sob essa hipótese razoável, prova-se que quebrar o esquema QC-MDPC não é mais fácil do que resolver o problema da decodificação por síndrome, que é conhecidamente $\mathcal{NP}$-difícil, e conjecturado tipicamente difícil. Porém, essa boa redução de segurança, baseada em problemas aceitos como difíceis, não leva em conta o fato de o algoritmo de decriptação falhar com certa probabilidade, em torno de 0.1% [MOG15]. Como a falha gera um pedido ao remetente, um ataque de reação é possível.

3.3 Ataque de reação aos códigos QC-MDPC

Na AsiaCrypt 2016, Guo, Johansson, e Stankovski [GJS16] mostraram um ataque de reação para a recuperação de chave para o esquema de McEliece instanciado com códigos QC-MDPC. O ataque é inteiramente derivado da obervação de que a probabilidade de haver um erro de decodificação de um vetor $\boldsymbol{c} = \boldsymbol{m} \oplus \boldsymbol{e}$ é menor quando o a primeira linha do primeiro bloco cíclico $\boldsymbol{h}_0$, e o vetor de erro $\boldsymbol{e}$, compartilham algumas propriedades. Dessa forma, uma atacante Eva envia desafios de decodificação com erros artificialmente construídos, de modo a capturar informações estruturais sobre a $\boldsymbol{h}_0$. Depois, com essas informações, Eva reconstroi o vetor $\boldsymbol{h}_0$, e com H_0 e a chave pública, a atacante recupera a chave H completamente.

3.3.1 Descrição do ataque

A informação sobre $\boldsymbol{h}_0$ que o ataque é capaz de recuperar o seu espectro, que é o conjunto das menores distâncias cíclicas entre as suas entradas não nulas. A definição formal de distância cíclica é dada a seguir.

Definição 3.3.1 (Distância cíclica e par d-distante)**.** Seja e um vetor de comprimento r com elementos. A *distância cíclica* entre duas posições i e j o único inteiro positivo d menor do que $\lfloor r/2 \rfloor$ tal que $d \equiv j - i \mod r$ ou $d \equiv i - j \mod r$.

Dizemos que um par de índices (s_1, s_2) é *d-distante* se sua distância cíclica é d.

Formalmente, o espectro é definido a seguir.

Definição 3.3.2 (Espectro)**.** Seja $v = [v_1 \ldots v_r]$ um vetor de comprimento r com elementos em $\mathbb{F}_2$. O espectro de v, denotado por $\sigma(v)$, é o conjunto

$$\sigma(v) = \{d \in \mathbb{Z} : \text{existe um par } d\text{-distante de entradas não nulas em } v\}.$$

Como exemplo, considere o vetor $[1\,0\,0\,1\,0\,1\,0\,0\,0\,1]$. Seu espectro é o conjunto $\{1, 2, 3, 4, 5\}$. Note que uma distância cíclica entre elementos de um vetor de comprimento r sempre é igual ou inferior a $\lfloor r/2 \rfloor$, e o número máximo de elementos distintos de seu espectro é igual a $\lfloor r/2 \rfloor$.

O ataque consiste em duas fases: a de recuperação do espectro de $\boldsymbol{h}_0$, e a de reconstrução de $\boldsymbol{h}_0$ a partir de seu espectro. A ideia é enviar mensagens encriptadas com erros artificiais de modo a testar cada uma das possíveis distâncias, $d = 1, \ldots, \lfloor r/2 \rfloor$, e usar a taxa de falha do decodificador para descobrir se cada d está ou não em $\sigma(\boldsymbol{h}_0)$. Para testar cada distância d, os autores introduzem um conjunto de erros chamado Ψ_d, definido a seguir.

Definição 3.3.3 (Conjunto Ψ_d)**.** Suponha que queremos obter informações sobre uma chave privada de um código de comprimento n, codimensão $r = n/2$, e que corrige t erros. O *conjunto de testes da distância d* é definido como o conjunto

$$\Psi_d = \left\{ (\boldsymbol{e} \mid \boldsymbol{0}) : \text{ e } e \text{ tem } \frac{t}{2} \text{ pares distintos de índices d-distantes} \right\},$$

em que o comprimento de $\boldsymbol{e}$ é igual a r.

Os autores do ataque observaram que para cada distância d, quando a encriptação usa erros em Ψ_d, a taxa de falha de decodificadores por *bit-flipping* tende a ser mais significativamente mais baixa quando d pertence ao espectro de $\boldsymbol{h}_0$ do que quando d não pertence ao espectro. Isso acontece pois, quando $d \in \sigma(\boldsymbol{h}_0)$, os erros em Ψ_d fazem com que a primeira iteração do algoritmo de *bit-flipping* tenha mais sucesso na primeira iteração. Na próxima seção, analisamos o ataque um pouco mais detalhadamente.

3.3.2 Explicação do ataque

Como o algoritmo de *bit-flipping* tipicamente termina em poucas iterações, o seu desempenho na primeira iteração é crítico para que ele possa corrigir o padrão de erros. Nesse algoritmo, cada variável v_j tem um contador associado f_j. Este contador é incrementado quando v_j faz parte de uma equação de paridade não satisfeita. Analisando a primeira iteração do algoritmo na decodificação de um vetor $\boldsymbol{c} = \boldsymbol{m} \oplus \boldsymbol{e} = [v_1, v_2, \ldots, v_n]$, temos que a síndrome de $\boldsymbol{c}$ é

$$\boldsymbol{s} = \boldsymbol{c}H^T = \boldsymbol{e}H^T = [s_1, s_2, \ldots, s_r],$$

onde cada s_i é da forma

$$s_i = \sum_{j=1}^{n} h_{ij} e_j.$$

Dessa forma, se $s_i = 0$, a equação foi satisfeita, e nenhum contador é incrementado. E se $s_i = 1$, são incrementados os contadores de cada variável que aparece na equação i.

Podemos analisar a primeira iteração de decodificação um pouco mais detalhadamente, ao considerar quantos contadores são incrementados corretamente. Para simplificar a escrita, chame de λ_j o número de colunas de e tais que $h_{ij} = 1$, ou seja $\lambda_j = \#\{j : h_{ij} e_j = 1\}$. Note que $\lambda_j = s_j \mod 2$, mas em geral $\lambda_j \neq s_j$, então s_j só nos dá informação sobre a paridade de λ_j.

Se $\lambda_j = 0$ temos que a equação j não poderia identificar nenhum bit incorreto, e os contadores das w variáveis na equação j não são modificados. Por outro lado, se $\lambda_j = 1$, temos que a equação j captura apenas 1 bit incorreto, mas incrementa todos os contadores das suas w variáveis, sendo que apenas 1 dos contadores deveria ser incrementado.

Dizemos que l contadores são *tratados corretamente* quando uma equação insatisfeita incrementa l contadores de variáveis que estavam em erro, ou quando uma equação satisfeita deixa de incrementar l contadores que não estavam em erro. Analogamente, dizemos que que l contadores são *tratados incorretamente* quando uma equação insatisfeita deixa de incrementar l contadores de variáveis que não estavam em erro, ou quando uma equação satisfeita deixa de incrementar l contadores que estavam em erro. É claro, pela definição, que a soma do número de contadores tratados correta e incorretamente numa equação é sempre w. A Tabela 3.3.1 mostra a relação entre λ_j e o tratamento dos contadores.

λ_j	contadores tratados corretamente	contadores tratados incorretamente
0	w	0
1	1	$w-1$
2	$w-2$	2
3	3	$w-3$
$\vdots$	$\vdots$	$\vdots$

Tabela 3.3.1: *Relação entre λ_j e o número de contadores tratados correta ou incorretamente.*

Assim, para o decodificador cometer menos erros na primeira iteração, é interessante que λ_j seja um número par pequeno, ou um ímpar grande. Em particular, queremos aumentar a probabilidade de λ_j ser 0 com alta probabilidade, e diminuir a probabilidade de λ_j ser 1. O ataque constroi erros que aumentam a probabilidade de a primeira iteração ser boa se os erros e a chave compartilharem distâncias entre as entradas não nulas.

Para os testes de decodificação, os erros serão retirados aleatoriamente dos conjuntos Ψ_d. Quando selecionamos erros em Ψ_d, e existe ao menos um par d distante em $\boldsymbol{h}_0$, então garantidamente imporemos um par de uns em ao menos $t/2$ equações, como ilustra a Figura 3.3.1. . Essa mudança na distribuição de λ_j é detectável por meio de simulações, e a mudança relatada pelos autores está na Tabela 3.3.2. Com a probabilidade de λ_j ser 0 ser ligeiramente aumentada%, e a probabilidade de λ_j ser 1 ser ligeiramente diminuida, quando $\boldsymbol{h}_0$ contém um par d-distante, a probabilidade de a primeira iteração tratar corretamente mais contadores é maior.

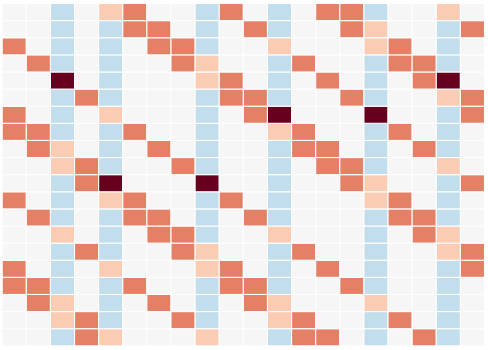


Figura 3.3.1: *Erro em Ψ_4 sobreposto a um bloco cíclico.*

Por meio de simulações, os autores encontraram a distribuição das taxas de erros, quando

λ_j	Probabilidade	
	$\boldsymbol{h}_0$ não contém nem um par d-distante	$\boldsymbol{h}_0$ contém ao menos um par d-distante
0	0.4485	0.4534
1	0.3663	0.3602
≥ 2	0.1852	0.1864

Tabela 3.3.2: *Diferentes distribuiçãoes de λ_j quando os erros são tirados de Ψ_d.*

há ou não distâncias d-distantes. Eles sugerem que as distribuições podem ser aproximadas para distribuições normais, com certas médias e variâncias mostradas na Tabela 3.3.3. Assim, por meio de testes de hipótese, é possível classificar uma distância como existente ou não em $\boldsymbol{h}_0$.

Existe par d-distante em $\boldsymbol{h}_0$?	Taxa média de erro	Desvio padrão
Sim	0.0044099	0.00003868
Não	≤ 0.0009116	≤ 0.00001304

Tabela 3.3.3: *Taxas médias de falhas de decodificação quando há ou não pares d-distantes em $\boldsymbol{h}_0$.*

O Algoritmo 5 formaliza os passos que vimos até agora para encontrar o conjunto de distâncias existentes em $\boldsymbol{h}_0$.

Algoritmo 5: Cálculo do conjunto de distâncias em $\boldsymbol{h}_0$ como em [GJS16]

Entrada: Parâmetros n, r, w, t do código QC-MDPC.
$\mathcal{D}$ o oráculo de reação do decodificador.
M o número de desafios de decodificação para cada distância.

Saída: $\sigma(\boldsymbol{h}_0)$ o espectro de $\boldsymbol{h}_0$.

```
1  início
2      para cada distância d = 1, 2, ..., ⌊r/2⌋ faça
3          p ← 0
4          para cada iteração i = 1, 2, ..., M faça
5              e ←$ Ψ_d
6              c ←$ C
7              se D(c ⊕ e) = 1 então
8                  p ← p + 1
9          Decida se d ∈ σ(h_0) com base na proporção p/M
10     devolva σ(h_0)
```

E finalmente, depois de calculado o conjunto de distâncias existentes em $\boldsymbol{h}_0$, fazemos uma busca em profundidade com as distâncias encontradas, usando o Algoritmo recursivo 6. Essa busca encontra $\boldsymbol{h}_0$, a menos de um *shift* circular, o que não faz diferença, já que qualquer *shift* gera uma matriz equivalente a H. Também é fácil testar se a $\boldsymbol{h}_0$ é mesmo o vetor, bastando recuperar H por meio de álgebra linear simples, e ver se G é o núcleo de H.

Primeiro, tome a menor distância do espectro, chamada d_0. Depois, construa um conjunto com posições em potencial de $\boldsymbol{h}_0$, chamado $\mathcal{B}$. O conjunto $\mathcal{B}$ contém o subconjunto das posições possíveis $\{1, 2, \ldots, r-1\}$ dos elementos para os quais as distâncias até a posição 0 e até a posição d_0 estejam

no espectro. Ou seja:

$$\mathcal{B} = \{b : \text{dist}_c(b, 0) \in \sigma(\boldsymbol{h}_0) \wedge \text{dist}_c(b, d_0) \in \sigma(\boldsymbol{h}_0)\}.$$

Sendo $b_0 < b_1 < b_2 < \ldots$ uma ordenação do conjunto $\mathcal{B}$, faça uma busca em profundidade na árvore da Figura 3.3.2. Quando a distância de um nó a um de seus ancestrais não existir no espectro de $\boldsymbol{h}_0$, busque o próximo nó filho de seu pai, se existir, ou suba de nível, recursivamente. Apesar de parecer uma busca demorada, os autores notam que é esperado que a árvore passe a ter poucos filhos rapidamente, devido à baixa probabilidade de várias distâncias em comum estarem no espectro.

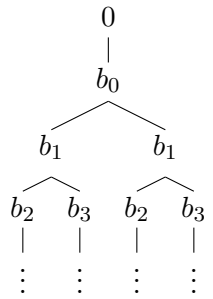


Figura 3.3.2: *Árvore de busca de* $\boldsymbol{h}_0$.

Algoritmo 6: Recuperação de $\boldsymbol{h}_0$.

Entrada: Parâmetros n, r, w, t do código QC-MDPC.
O conjunto de distâncias $\sigma(\boldsymbol{h}_0)$.
Chave parcial $\boldsymbol{h}_0$.
Nível da recursão l.

Saída: O vetor recuperado $\boldsymbol{h}_0$ ou $\perp$ se não houver um vetor com espectro $\sigma(\boldsymbol{h}_0)$.

```
1  início
2      se l = w então
3          devolva h_0
4      para todo bit j = 1, ..., n faça
5          se todas as distâncias entre bits não nulos e j estiver em σ(h_0) então
6              Adicione bit j a h_0
7              Chamada recursiva(σ(h_0), h_0, l + 1)
8              se h_0 for a chave secreta então
9                  devolva h_0
10             Remova bit i de h_0
11     devolva ⊥
```

Com pequenas modificações, o ataque pode funcionar até para conversões IND-CCA2 do esquema de McEliece, embora menos eficientemente. Claro que apenas a primeira fase é afetada quando se usa conversões de segurança.

Restam alguns ramos de investigação interessantes como a possibilidade de se recuperar a $\boldsymbol{h}_0$ mesmo com erros no conjunto de distâncias, ou saber se o ataque funciona com decodificadores por decisões suaves.

Capítulo 4

Resultados Preliminares

A estimação do espectro sugerida pelos autores do ataque usa um grande número de pedidos de decodificação a um oráculo decriptador para estimar as taxas de falha de decodificação para cada distância. Idealmente gostaríamos de poder classificar toda distância como pertencente ou não ao espectro. Usando o algoritmo de *bit-flipping* de Gallager, os autores do ataque conseguem fazer isso através de 100000 testes para cada distância. Porém quando é usada a variante do *bit-flipping* de Misoczki, que tem menor taxa de falha, 100000 testes não são suficientes para classificar cada distância, como pode ser observado na Figura 4.0.1. Note como há dois picos, um próximo à taxa 0.0006, e outro próximo à taxa 0.0009, que devem corresponder aos centros das distribuições das taxas de falha para distâncias dentro e fora do espectro, respectivamente. Porém não há um ponto de separação entre distâncias dentro e fora do espectro.

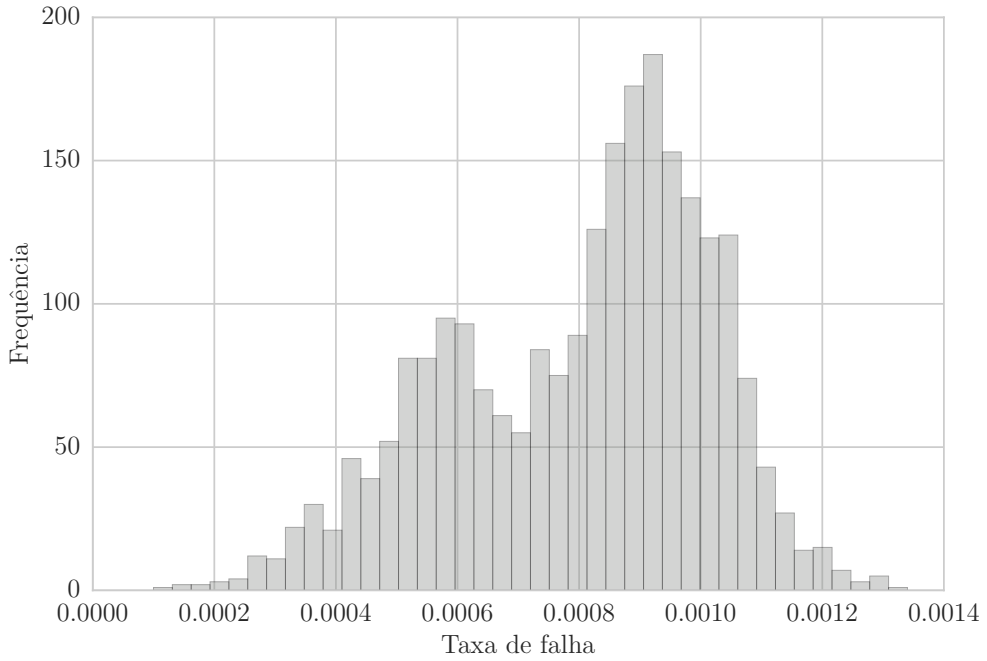


Figura 4.0.1: *Histograma das taxas de falha de decodificação estimadas com 100000 testes para cada distância, usando um código QC-MDPC gerado aleatoriamente.*

Duas opções possíveis para tratar da não separação completa das distâncias dentro e fora do espectro são:

- fazer um número maior de testes para as distâncias em regiões intermediárias até obter separação;
- usar um algoritmo de recuperação de chave que lide com os erros na recuperação do espectro.

A primeira opção é a mais conservadora. Porém, pode-se demorar muito até a obtenção de uma separação completa. Usando o *bit-flipping* de Misoczki, as simulações dos 100000 testes para cada uma das 2400 distâncias possíveis de $\boldsymbol{h}_0$ de códigos (9602, 4801, 90)-QC-MDPC demoram cerca de 48 horas num computador com 8 *threads*. Como o *bit-flipping* de Misoczki falha cerca de 10 vezes menos do que o de Gallager, esperamos que haja separação com 10 vezes mais testes para cada distância. Isso é um problema pois faria a simulação demorar em torno de 20 dias para cada código. Então vamos explorar uma possível modificação do ataque para lidar com erros no espectro recuperado.

Nossa primeira observação é que o algoritmo de reconstrução da chave a partir do espectro tem bom desempenho quando há até cerca de 200 entradas erradas no espectro. Dessa forma não é necessário haver a separação completa entre distâncias dentro e fora do espectro.

Nossa segunda observação diz respeito a como decidir quais entradas serão consideradas como pertencentes ao espectro. Para isso, sugerimos usar o modelo de mistura gaussiana para achar as distribuições Normais que melhor aproximam as taxas de falha observadas para cada multiplicidade. Uma vez em posse dessas distribuições, podemos calcular parâmetros que otimizam o desempenho do algoritmo de recuperação da chave a partir do espectro.

Duas das vantagens de usar esse modelo em relação ao ataque original são:

- não é necessário conhecer a priori a distribuição das taxas de falha de decodificação para cada decodificador;
- o tempo de recuperação é significativamente mais rápido, pois são necessários bem menos testes de decodificação, que são a fase mais demorada do ataque.

Naturalmente, ambas as vantagens são úteis para estender o ataque ao caso do esquema convertido para IND-CCA2. Com esse modelo, também fomos capazes de atacar o QC-MDPC McEliece usando decodificadores por decisões suaves, o que foi conjecturado mas não mostrado pelos autores originais do ataque.

4.1 Reconstrução da chave a partir do espectro

Queremos modificar o algoritmo de reconstrução de chave para que consiga reconstruir a chave mesmo que o espectro esteja parcialmente correto. Então começamos analisando o desempenho do algoritmo original.

Os autores declararam que a execução do algoritmo demora em média 144 segundos, mas não dão uma análise sobre o tempo de execução[1]. Implementamos o algoritmo com algumas pequenas otimizações e fizemos testes de desempenho com 1000 espectros de $\boldsymbol{h}_0$ de códigos aleatórios. Destes testes, 75% das chaves foram recuperadas em menos de 5 segundos, e 99.7% das entradas foram recuperadas em menos de 90 segundos. A distribuição de frequências para os tempos de execução dos testes pode ser vista na Figura 4.1.1.

[1] Os autores dizem “alguns minutos” no artigo. Mas o tempo médio de 144 segundos pode ser visto no vídeo da apresentação na Asiacrypt em https://www.youtube.com/watch?v=tKvDdGLJLZc.

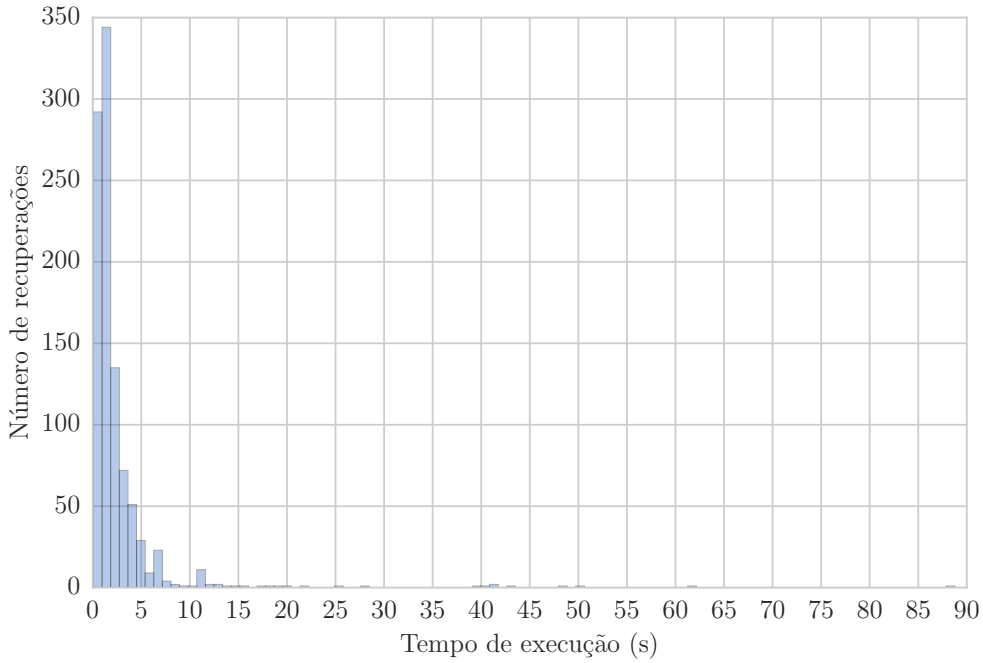


Figura 4.1.1: *Histograma do tempo de recuperação pelo espectro de $\boldsymbol{h}_0$ para 1000 códigos aleatórios.*

Pela descrição do Algoritmo 6, é fácil ver que o algoritmo não funciona quando faltam distâncias no espectro. Porém o algoritmo lida bem quando o espectro passado tem, além de todas as entradas do espectro real, algumas entradas incorretas adicionadas. A Tabela 4.1.1 mostra resultados de uma simulação simples do algoritmo de reconstrução para um espectro fixo com 821 distâncias corretas, mas certos números de entradas erradas adicionais. Dessa forma, mesmo tomando um conjunto que contenha o espectro, mas que também possua entre 164 e 205 entradas fora do espectro, ainda teremos um bom tempo de execução. Este é o principal fato usado para baixar o número de testes de decodificação para cada distância.

Número distâncias erradas inseridas	Tempo de execução
0	4s
41	11s
82	22s
123	1m41s
164	4m56s
205	18m24s
246	47m58s
287	5h18m7s

Tabela 4.1.1: *Tempo de execução do algoritmo de reconstrução quando são inseridas distâncias erradas no espectro.*

A análise da complexidade da reconstrução dada pelos autores explica por que o algoritmo é capaz de lidar bem com distâncias erradas inseridas no espectro. O motivo é que, a cada nível descido na árvore de busca como a da Figura **??**, o número de filhos de cada nó fica exponencialmente menor. Assim, para o bom funcionamento do algoritmo de recuperação, precisamos minimizar o número de candidatos a bits da chave.

Nossa proposta é que o algoritmo de recuperação de chave rode em duas fases:

1. Reconstrução da chave parcial usando apenas distâncias que pertencem ao espectro com maior probabilidade;

2. Completação da chave parcial usando também as distâncias que têm menor probabilidade de pertencer ao espectro.

A ideia é que erros no espectro são péssimos para a Fase 1, porém não tão ruins para a Fase 2. Isso, pois o algoritmo recupera grande parte da chave rapidamente na Fase 1, com poucos erros no espectro. Depois é esperado que poucas posições sejam válidas, mesmo com um número até que grande de distâncias erradas no espectro.

Algoritmo 7: Reconstrução de $\boldsymbol{h}_0$.

Entrada: $\boldsymbol{h}_0$ índices não nulos da chave parcialmente recuperada a cada chamada recursiva; na primeira chamada é o conjunto vazio.
B_0 índices não nulos em potencial de $\boldsymbol{h}_0$ com maior probabilidade.
S_0 distâncias pertencentes ao espectro com maior probabilidade.
B_1 índices não nulos em potencial de $\boldsymbol{h}_0$ com menor probabilidade.
S_1 distâncias pertencentes ao espectro com menor probabilidade.
δ_b estimativa conservadora de $|\boldsymbol{h}_0 - B_0|$.
δ_s estimativa conservadora de $|\sigma(\boldsymbol{h}_0) - S_0|$.

Saída: O vetor recuperado por referência em $\boldsymbol{h}_0$, ou $\perp$ se não encontrar o vetor.

```
1  início
2      para todo j ∈ B_0 maior que o último índice não nulo de h_0 faça
3          se as distâncias entre j e cada índice não nulo de h_0 estão em S_0 então
4              Adicione j a h_0
5              se ChamadaRecursiva(h_0, l + 1, B_0, S_0, B_1, S_1) = ⊤ então
6                  devolva ⊤
7              Remova j de h_0
8      se w(h_0) ≥ w/2 − δ_b − δ_s então
9          devolva Fase2(h_0, B_0, S_0, B_1, S_1, ∅, δ_s)
10     devolva ⊥
```

É fácil ver que a nossa proposta é uma generalização do algoritmo de recuperação de chave original. Para isso, tome $B_1 = B_0 = S_1 = S_0 = \sigma(\boldsymbol{h}_0)$, o que faz $\delta_s = \delta_b = 0$. Agora vamos fazer algumas observações sobre os parâmetros.

1. Seja p o número de posições não nulas de $\boldsymbol{h}_0$ que devem ser descobertos na Fase 1. Considere I o conjunto de índices não nulos do *shift* circular de $\boldsymbol{h}_0$ que maximiza $|I \cap B_0|$. Então uma cota inferior para p é

$$p \geq w/2 - |I - B_0| - |\sigma(\boldsymbol{h}_0) - S_0|. \tag{4.1.1}$$

   Isso pois $|I - B_0|$ é o número de índices que fazem parte de $\boldsymbol{h}_0$, mas que j não valerá na Fase 1. E $|\sigma(\boldsymbol{h}_0) - S_0|$ é uma cota superior para o número de índices j válidos que serão descartados incorretamente pelo fato de a distância entre o índice j e algum outro índice de $\boldsymbol{h}_0$ parcial não estar em S_0. Isso justifica o limite $\mathrm{w}(\boldsymbol{h}_0) \geq w/2 - \delta_b - \delta_s$ na Fase 1 do algoritmo.

2. Na Fase 2, o conjunto D consiste de todas as distâncias de S_1 usadas no espectro de $\boldsymbol{h}_0$. É permitido o uso máximo de δ_s distâncias em S_1, pois esse é o número de entradas no espectro que não devem ter sido capturadas por S_0.

Nas próximas seções, mostramos nossa proposta para escolher os conjuntos S_0, S_1, B_0, e B_1, e como estimar os parâmetros δ_s e δ_b.

Algoritmo 8: Reconstrução de $\boldsymbol{h}_0$ - FASE2

Entrada: $\boldsymbol{h}_0$ índices não nulos da chave parcialmente recuperada a ser completada.
B_0 índices não nulos em potencial de $\boldsymbol{h}_0$ com maior probabilidade.
S_0 distâncias pertencentes ao espectro com maior probabilidade.
B_1 índices não nulos em potencial de $\boldsymbol{h}_0$ com menor probabilidade.
S_1 distâncias pertencentes ao espectro com menor probabilidade.
D conjunto de distâncias em S_1 usadas; inicialmente é $\varnothing$.
δ_s estimativa conservadora de $|\sigma(\boldsymbol{h}_0) - S_0|$.

Saída: O vetor recuperado por referência em $\boldsymbol{h}_0$, ou $\perp$ se não encontrar o vetor.

```
1  início
2      se w(h_0) = w então
3          se h_0 for a chave secreta então
4              devolva ⊤
5          senão
6              devolva ⊥
7      para todo bit em potencial j ∈ B_0 ∪ B_1 faça
8          D_temp ← D
9          para cada bit b não nulo de h_0 faça
10             d ← distância circular mínima entre j e b
11             se d ∈ S_1 então
12                 D ← D ∪ d
13             senão se d ∉ S_0 então
14                 devolva ⊥
15         se |D| ≤ δ_s então
16             Adicione j a h_0
17             Chamada recursiva a FASE2(h_0, B_0, S_0, B_1, S_1, D, δ_s)
18             Remova j de h_0
19         D ← D_temp
20     devolva ⊥
```

4.2 Obtenção dos parâmetros para o algoritmo de reconstrução

Para estimar os parâmetros para o algoritmo de reconstrução de chave, precisamos de um modo de extrair a distribuição de probabilidade das taxas de falha para distâncias dentro e fora do espectro. Nossa proposta é usar o modelo de mistura gaussiana (GMM)-*Gaussian Mixture Model* - que considera que a distribuição final, como a da Figura 4.0.1, é uma soma distribuições Normais. Cada distribuição Normal representa uma multiplicidade possível das distâncias no espectro. Há algoritmos iterativos eficientes, como o de maximização de esperança, que permitem essa decomposição. Assim, chamando de $\mathcal{D}$ a distribuição de probabilidade para as taxas de falha dos testes para uma distância válida aleatória, o modelo por mistura gaussiana descreve $\mathcal{D}$ como

$$\mathcal{D} = \sum_{i=0}^{\max \mu[d]} \lambda_i \mathcal{N}_i.$$

Cada $\mathcal{N}_i$ é uma distribuição Normal associada à multiplicidade i, e cada λ_i é um número real no intervalo $[0, 1]$ chamado peso para a distribuição $\mathcal{N}_i$. Cada λ_i representa a probabilidade de uma distância válida escolhida aleatoriamente ter multiplicidade i, então a soma dos λ_i é igual a 1. Note que os λ_i são independentes do decodificador, enquanto os parâmetros de cada Normal $\mathcal{N}_i$ são altamente dependentes do decodificador. Idealmente, as Normais encontradas pelo GMM seriam as normais que melhor ajustam as taxas de falha observadas para distâncias com cada uma das multiplicidades possíveis, como ilustra a Figura 4.2.1.

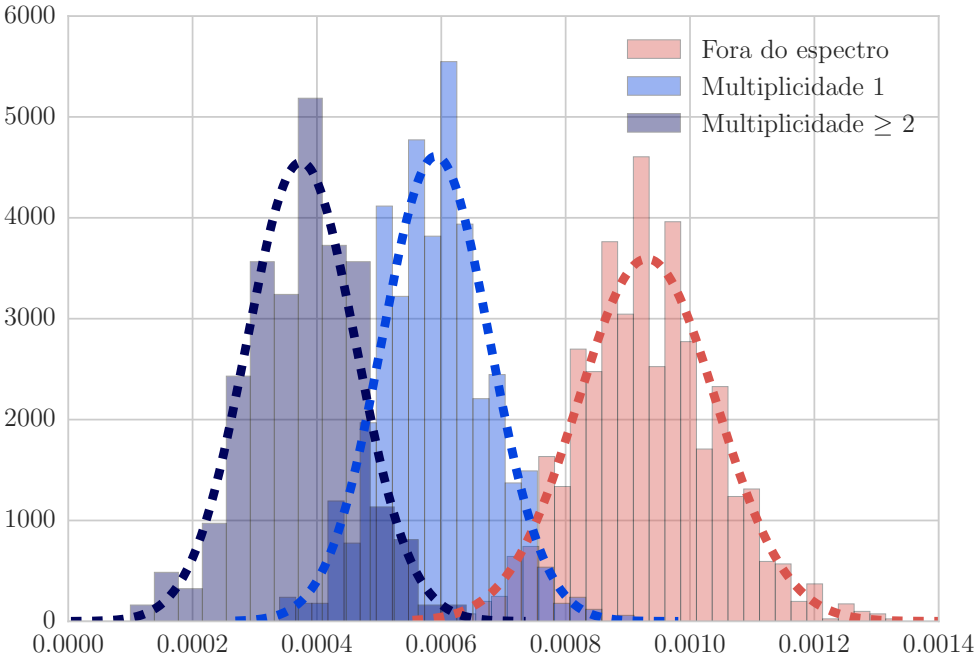


Figura 4.2.1: *Normais que melhor ajustam as taxas de falha observadas para cada multiplicidade. Aglomeramos as multiplicidades iguais ou superiores a 2 em apenas uma normal para que as figuras coubessem na mesma escala do gráfico.*

Porém a qualidade do ajuste de cada $\mathcal{N}_i$ é dependente do número distâncias com multiplicidade i. Para as multiplicidades mais comuns, como 0 e 1, o ajuste tende a funcionar bem. Mas para multiplicidades maiores, que são mais raras, o ajuste pode não convergir. A Tabela 4.2.1 gerada através de simulações pelos autores do ataque mostra as probabilidades de cada multiplicidade ocorrer num $\boldsymbol{h}_0$ de um código $9602, 4801, 90$-QC-MDPC. Note como apenas em torno de 1% das distâncias têm multiplicidade maior do que 3, o que para um código em que $\boldsymbol{h}_0$ tem peso[2] 45,

[2]Se $\boldsymbol{h}$ tem peso 90, o peso esperado de $\boldsymbol{h}_0$ é 45.

Multiplicidade (i)	Probabilidade (λ_i)	Acumulada ↓	Acumulada ↑
0	0.6589889	0.6589889	1.0000000
1	0.2748075	0.9337965	0.3410106
2	0.0573330	0.9911295	0.0662031
3	0.0079677	0.9990972	0.0088701
4	0.0008287	0.9999260	0.0009024
5	0.0000688	0.9999949	0.0000737
6	0.0000047	0.9999997	0.0000049
7	0.0000002	1.0000000	0.0000002

Tabela 4.2.1: *Probabilidades estimadas de ocorrerem as multiplicidades mais comuns em vetores de tamanho 4801 e peso 45, condizentes com $\boldsymbol{h}_0$ de um código que suporta um nível de segurança de 80.*

equivale a pouco menos de $0.01\binom{45}{2} \approx 10$ entradas.

Vemos duas alternativas para tratar da não convergência das Normais do GMM quando tentamos ajustar mais de duas Normais aos nossos dados:

1. ajustar todos os dados a apenas duas Normais, uma para distâncias fora do espectro (multiplicidade 0), e outra para distâncias dentro do espectro (multiplicidade igual ou superior a 1);

2. descartar dados de distâncias que provavelmente têm multiplicidade maior do que 1 para encontrar um bom ajuste para as duas Normais que representam apenas as multiplicidades 0 e 1.

A primeira alternativa parece razoável, porém tem o problema de sempre superestimar os dados na cauda direita da Normal das distâncias dentro do espectro. Isso acontece pela alta variância causada pelas baixas taxas de falha para distâncias de multiplicidades maiores do que 1, o que é ilustrado pela linha pontilhada verde na Figura 4.2.2. Isso é um problema sério para consultas sobre a inversa da densidade acumulada da Normal das distâncias de dentro do espectro. Por exemplo, a taxa de falha que deixa à esquerda 99.99% das distâncias dentro do espectro seria superestimado. E assim, o algoritmo de recuperação a partir do espectro teria que trabalhar com um número significativamente maior de entradas erradas do que se o ajuste fosse mais fino.

A segunda alternativa levanta a questão natural de como decidir quais dados descartar para obter um bons ajustes na maior parte dos casos. Note como aproximadamente 6.62% das distâncias têm multiplicidade igual ou superior a 2, pela Tabela 4.2.1. Assim, em torno de 3.31% dos dados com menores taxas de falha devem corresponder à cauda esquerda da distribuição das taxas de falha para distâncias de multiplicidade igual ou superior a 2. Nossa sugestão é descartar entre 2% e 3% das menores taxas de falha para fazer o ajuste. A melhor qualidade do ajuste para zonas intermediárias entre distâncias dentro e fora do espectro é ilustrada pela linha pontilhada azul da Figura 4.2.2.

Para gerar os ajustes da Figura 4.2.2, usamos o pacote mixtools [BCHY09] da linguagem R. E para facilitar a comparação dos ajustes, são mostrados também os histogramas das taxas de falha para os testes com distâncias dentro e fora do espectro. Lembre que os todos os ajustes em pontilhado são feitos sem acesso ao espectro de distâncias, então são derivados somente a partir da distribuição em cinza da Figura 4.0.1.

As seções seguintes mostram uma tentativa de formalizar as escolhas dos parâmetros do algoritmo de reconstrução da chave. A análise formalização será útil para estimar um bom número de testes para cada distância para que o algoritmo de reconstrução funcione bem. Porém na prática, o algoritmo mostrou bom desempenho mesmo quando a escolha de parâmetros é feita de forma extremamente conservadora, com parâmetros fixos para o pior caso esperado, em todos os desafios.

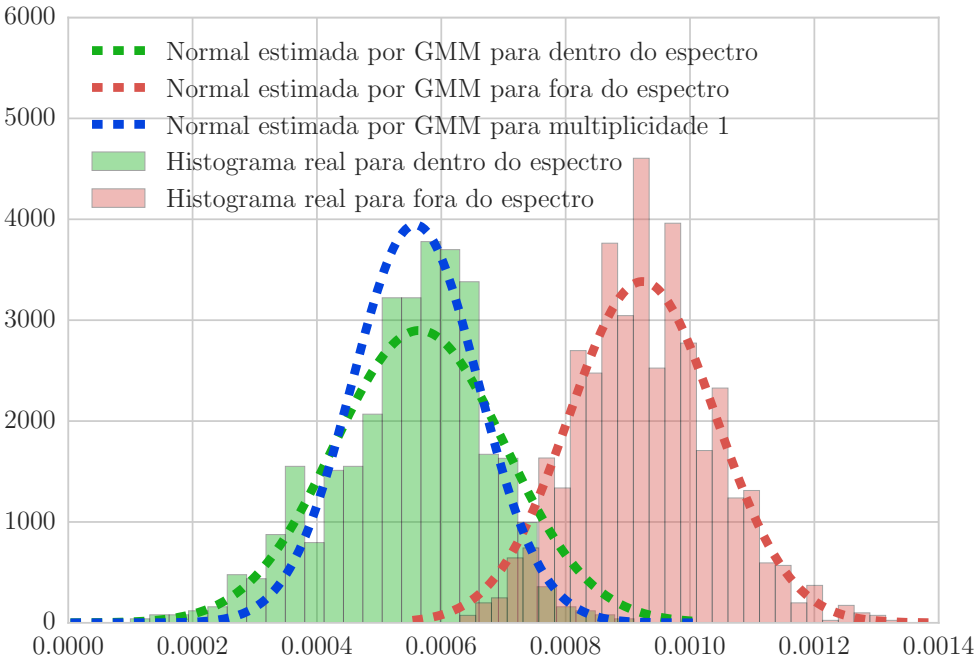


Figura 4.2.2: *Estimativa das distribuições das taxas de falha para distâncias dentro e fora do espectro.*

4.2.1 Os conjuntos S_0 e S_1

Em posse da distribuição $\mathcal{N}_1$, podemos fazer consultas sobre a inversa de sua função de densidade acumulada. Isso será útil para tomar o conjunto S_1 de modo que $S_0 \cup S_1$ contenham $\sigma(\boldsymbol{h}_0)$ com alta probabilidade. Mas lembre que o conjunto S_0 não pode ser muito grande por conta da performance da Fase 1 do algoritmo de reconstrução de chave.

Então sejam f a função inversa da densidade acumulada de $\mathcal{N}_1$, e $d_1 < d_2 < \ldots < d_{\lfloor r/2 \rfloor}$ uma ordenação das distâncias em função de suas taxas de falha $\pi[d]$, os conjuntos S_0 e S_1 são construídos como descrito a seguir.

- $S_0 = \{d_1, d_2, \ldots, d_{1000}\}$.
- $S_1 = \{d : \pi[d] \leq f(99.999\%)\} - S_0$.

A escolha de 1000 entradas para S_0 pode parecer artificial, mas note que é um número que tenta fazer com que a Fase 1 não demore muito. O índice i pode ser interpretado como o custo de usar uma distância do conjunto S_i na reconstrução da chave.

Também podemos estimar o número δ_s de distâncias do espectro $\sigma(\boldsymbol{h}_0)$ que o conjunto S_0 não captura. Essa quantidade será útil para o algoritmo de reconstrução. Por definição, temos

$$\delta_s = |\sigma(\boldsymbol{h}_0) - S_0| = |\sigma(\boldsymbol{h}_0)| - |\sigma(\boldsymbol{h}_0) \cap S_0|.$$

Seja a a densidade acumulada de $\mathcal{N}_1$ até o valor de $\pi[d_{1000}]$. O valor a é uma estimativa por GMM para a proporção de valores do espectro que B_0 captura. Então um estimador para o valor esperado do número esperado δ_s é

$$\overline{\delta_s} = \mathbb{E}(|\sigma(\boldsymbol{h}_0)|)(1 - a). \tag{4.2.1}$$

Note que o valor de $\mathbb{E}(|\sigma(\boldsymbol{h}_0)|)$ pode ser estimado usando uma tabela como a Tabela 4.2.1.

4.2.2 Os conjuntos B_0 e B_1

Queremos construir os conjuntos B_0 e B_1 de candidatos a índices não nulos de $\boldsymbol{h}_0$. Uma opção seria construir esses conjuntos nos baseando nos respectivos conjuntos S_0 e S_1 da seguinte forma. Considere d_1 como a distância que tem menor taxa de falha $\pi[d]$. A probabilidade d_1 pertencer ao espectro é alta, então suponha que d_1 pertence ao espectro[3].

Seja $\text{dist}_c(i,j)$ a distância circular mínima entre os índices i e j do vetor $\boldsymbol{h}_0$, então construa:

- $\overline{B}_0 = \{i = 1, 2, \ldots, r : \{\text{dist}_c(i,0), \text{dist}_c(i,d_1)\} \subset S_0\}$;
- $\overline{B}_1 = \{i = 1, 2, \ldots, r : \{\text{dist}_c(i,0), \text{dist}_c(i,d_1)\} \subset S_1\}$.

O problema desse modo é que não temos controle sobre o tamanho dos conjuntos, já que não temos nem uma métrica. Em particular,

Uma proposta que funciona melhor na prática é a seguinte. Para cada índice i, defina o par ordenado $B(i) = (\pi[\text{dist}_c(i,0)], \pi[\text{dist}_c(i,d_1)])$. E considere o conjunto dos pontos

$$B = \{B(i) : i = 1, 2, \ldots, r\}.$$

Considere uma ordenação dos índices i em relação à distância de $B(i)$ à origem, dada por $i_1 < i_2 < \ldots < i_r$. Note que como S_0, o conjunto B_0 também não pode ser muito grande. Então construa os conjuntos

- $B_0 = \{i_1, i_2, \ldots, i_{800}\}$;
- $B_1 = \{i : \|B(i) - (0,0)\| \leq R\}$, onde o raio R é escolhido de forma deixar alta a probabilidade de $B_0 \cup B_1$ conter as entradas não nulas de $\boldsymbol{h}_0$[4].

A probabilidade de um índice não nulo de $\boldsymbol{h}_0$ não pertencer a $B_0 \cup B_1$ pode ser estimada por:

$$\Pr(i \notin B_0 \cup B_1 | i \in \boldsymbol{h}_0) = \Pr(\|B(i) - (0,0)\| \leq R \,|\, \text{dist}_c(i,0) \text{ e } \text{dist}_c(i,d_1) \text{ seguem } \mathcal{N}_1).$$

De forma que essa probabilidade pode ser calculada usando algum método simples de Monte Carlo. Como um que gera pontos (X, Y), tais que $X, Y \sim \mathcal{N}_1$, e calcula a proporção de pontos que cai fora do círculo de raio R com centro na origem.

Similarmente ao final da seção anterior, podemos estimar o número δ_b de entradas do conjunto de posições não nulas de $\boldsymbol{h}_0$ que não devem ser capturadas por B_0. Seja q a probabilidade de um índice não nulo de $\boldsymbol{h}_0$ qualquer não pertencer a B_0. Essa probabilidade pode ser caculada por um método de Monte Carlo parecido com o descrito acima, usando como raio R a distância entre $B(i_{800})$ e a origem. Assim δ_b segue uma distribuição binomial com parâmetros q e $w/2 = 45$, e um estimador para o valor esperado de δ_b é

$$\overline{\delta_b} = q\frac{w}{2}. \tag{4.2.2}$$

4.2.3 Os parâmetros $\overline{\delta_s}$ e $\overline{\delta_b}$

A quantidade $w/2 - \overline{\delta_b} - \overline{\delta_s}$ tenta captar o momento em que se deve ir da Fase 1 para a Fase 2 do algoritmo de reconstrução de chave. A ideia é que o algoritmo rode a Fase 1 pelo máximo possível de iterações, e vá para a Fase 2 quando tiver descoberto um número de entradas não nulas de $\boldsymbol{h}_0$ próximo do máximo possível, dados B_0 e S_0.

Assim podemos passar para o algoritmo os estimadores $\overline{\delta_b}$ e $\overline{\delta_s}$ como os das Equações 4.2.2 e 4.2.1. Ou melhor do que isso. Como temos as distribuições de δ_b e δ_s, podemos usar estimadores mais conservadores, que superestimam δ_b e δ_s com alta probabilidade, para que nossa cota inferior seja atingida na Fase 1 com alta probabilidade.

[3] Essa suposição pode parecer forte, mas nosso ataque costuma rodar rapidamente. Então, se estiver demorando, digamos, mais do que 40 minutos, podemos recomeçar o ataque tomando o d_2 no lugar de d_1, e assim por diante.

[4] Ou de um *shift* de $\boldsymbol{h}_0$.

4.3 Desempenho na prática

Fizemos testes de execução do algoritmo de reconstrução de chave contra 8 desafios diferentes. Os desafios consistem do resultado de $M = 100000$ testes de decodificação para cada possível distância de $\boldsymbol{h}_0$ de um código $(9602, 4801, 90)$-QC-MDPC, que corresponde a um nível de segurança de 80. Todos os códigos foram gerados aleatoriamente, a menos da restrição de que o peso de $\boldsymbol{h}_0$ foi imposto em 45, como no ataque original.

Mesmo tendo que lidar com erros no espectro, o algoritmo de reconstrução teve desempenho comparável ao ataque original. O tempo de execução médio foi de 173 segundos, e a distribuição dos tempos de execução pode ser vista na Tabela 4.3.1.

Número do desafio	Tempo de execução (s)
1	30.14
2	432.61
3	46.10
4	114.15
5	176.47
6	313.73
7	3.827
8	264.24

Tabela 4.3.1: *Tempo de execução do ataque para alguns desafios.*

Como a geração de um desafio custa 2 dias, infelizmente até agora só temos 8 desafios. Porém os resultados são motivadores, e imaginamos que seja possível usar o ataque usando $M = 50000$, que equivale à metade das tentativas por distância usadas para gerar estes desafios.

Referências Bibliográficas

[ABB+] Daniel Augot, Lejla Batina, Daniel J Bernstein, Joppe Bos, Johannes Buchmann, Wouter Castryck, O Dunkelmann, Tim Güneysu, Shay Gueron, Andreas Hülsing et al. Initial recommendations of long-term secure post-quantum systems (2015). *URL: https://pqcrypto. eu. org/docs/initial-recommendations. pdf. Citations in this document*, 16. 2

[BBD09] Daniel J Bernstein, Johannes Buchmann e Erik Dahmen. *Post-quantum cryptography*. Springer Science & Business Media, 2009. 1

[BCGO09] Thierry P Berger, Pierre-Louis Cayrel, Philippe Gaborit e Ayoub Otmani. Reducing key length of the mceliece cryptosystem. Em *Progress in Cryptology–AFRICACRYPT 2009*, páginas 77–97. Springer, 2009. 2

[BCHY09] Tatiana Benaglia, Didier Chauveau, David Hunter e Derek Young. mixtools: An r package for analyzing finite mixture models. *Journal of Statistical Software*, 32(6):1–29, 2009. 31

[BCS13] Daniel J Bernstein, Tung Chou e Peter Schwabe. Mcbits: fast constant-time code-based cryptography. Em *International Workshop on Cryptographic Hardware and Embedded Systems*, páginas 250–272. Springer, 2013. 1, 10

[BMVT78] Elwyn R Berlekamp, Robert J McEliece e Henk CA Van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24(3):384–386, 1978. 1, 7

[BSC16] Marco Baldi, Paolo Santini e Franco Chiaraluce. Soft mceliece: Mdpc code-based mceliece cryptosystems with very compact keys through real-valued intentional errors. páginas 795–799, 2016. 2, 4, 15

[CFS01] Nicolas T Courtois, Matthieu Finiasz e Nicolas Sendrier. How to achieve a mceliece-based digital signature scheme. Em *International Conference on the Theory and Application of Cryptology and Information Security*, páginas 157–174. Springer, 2001. 1

[FGUO+13] Jean-Charles Faugere, Valérie Gauthier-Umana, Ayoub Otmani, Ludovic Perret e Jean-Pierre Tillich. A distinguisher for high-rate mceliece cryptosystems. *IEEE Transactions on Information Theory*, 59(10):6830–6844, 2013. 2, 7

[FO99] Eiichiro Fujisaki e Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. Em *Annual International Cryptology Conference*, páginas 537–554. Springer, 1999. 12

[FOP+16] Jean-Charles Faugere, Ayoub Otmani, Ludovic Perret, Frédéric De Portzamparc e Jean-Pierre Tillich. Structural cryptanalysis of mceliece schemes with compact keys. *Designs, Codes and Cryptography*, 79(1):87–112, 2016. 2

[FOPT10] Jean-Charles Faugere, Ayoub Otmani, Ludovic Perret e Jean-Pierre Tillich. Algebraic cryptanalysis of mceliece variants with compact keys. Em *Advances in Cryptology–Eurocrypt 2010*, páginas 279–298. Springer, 2010. 2

[Gab05] Philippe Gaborit. Shorter keys for code based cryptography. Em *Proceedings of the 2005 International Workshop on Coding and Cryptography (WCC 2005)*, páginas 81–91, 2005. 2

[Gal62] Robert Gallager. Low-density parity-check codes. *IRE Transactions on information theory*, 8(1):21–28, 1962. 4, 15

[GJ16] Qian Guo e Thomas Johansson. A p-ary mdpc scheme. Em *Information Theory (ISIT), 2016 IEEE International Symposium on*, páginas 1356–1360. IEEE, 2016. 15

[GJS16] Qian Guo, Thomas Johansson e Paul Stankovski. A key recovery attack on mdpc with cca security using decoding errors. Em *22nd Annual International Conference on the Theory and Applications of Cryptology and Information Security (ASIACRYPT), 2016*, 2016. 2, 19, 22

[HPS98] Jeffrey Hoffstein, Jill Pipher e Joseph H Silverman. Ntru: A ring-based public key cryptosystem. Em *International Algorithmic Number Theory Symposium*, páginas 267–288. Springer, 1998. 2

[HVMG13] Stefan Heyse, Ingo Von Maurich e Tim Güneysu. Smaller keys for code-based cryptography: Qc-mdpc mceliece implementations on embedded devices. Em *Cryptographic Hardware and Embedded Systems-CHES 2013*, páginas 273–292. Springer, 2013. 18

[KI01] Kazukuni Kobara e Hideki Imai. Semantically secure mceliece public-key cryptosystems-conversions for mceliece pkc. Em *International Workshop on Public Key Cryptography*, páginas 19–35. Springer, 2001. 1, 4, 10, 12, 13

[MB09] Rafael Misoczki e Paulo SLM Barreto. Compact mceliece keys from goppa codes. Em *Selected Areas in Cryptography*, páginas 376–392. Springer, 2009. 2

[McE78] R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. *Deep Space Network Progress Report*, 44:114–116, 1978. 1

[Mil86] V.S. Miller. Use of elliptic curves in cryptography. *Advances in Cryptology (CRYPTO85)*, páginas 417–426, 1986. 1

[MOG15] Ingo Von Maurich, Tobias Oder e Tim Güneysu. Implementing qc-mdpc mceliece encryption. *ACM Transactions on Embedded Computing Systems (TECS)*, 14(3):44, 2015. 19

[MTSB13] Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier e Paulo SLM Barreto. Mdpc-mceliece: New mceliece variants from moderate density parity-check codes. Em *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*, páginas 2069–2073. IEEE, 2013. 2, 15, 18

[MVOV96] Alfred J Menezes, Paul C Van Oorschot e Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 1996. 1

[Nie86] Harald Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *PROBLEMS OF CONTROL AND INFORMATION THEORY-PROBLEMY UPRAVLENIYA I TEORII INFORMATSII*, 15(2):159–166, 1986. 1

[OTD10] Ayoub Otmani, Jean-Pierre Tillich e Léonard Dallot. Cryptanalysis of two mceliece cryptosystems based on quasi-cyclic codes. *Mathematics in Computer Science*, 3(2):129–140, 2010. 2

[Pat75] Nicholas J Patterson. The algebraic decoding of goppa codes. *Information Theory, IEEE Transactions on*, 21(2):203–207, 1975. 4

[Poi00] David Pointcheval. Chosen-ciphertext security for any one-way cryptosystem. Em *International Workshop on Public Key Cryptography*, páginas 129–146. Springer, 2000. 12

[RL09] William Ryan e Shu Lin. *Channel codes: classical and modern*. Cambridge University Press, 2009. 4

[RLR78] L. M. Adleman R. L. Rivest, A. Shamir. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978. 1

[S+00] Hung-Min Sun et al. Further cryptanalysis of the mceliece public-key cryptosystem. *IEEE communications letters*, 4(1):18–19, 2000. 4

[Sho97] P. W. Shor. Polynomial time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1481–1509, 1997. 1

[SMR00] Amin Shokrollahi, Chris Monico e Joachim Rosenthal. Using low density parity check codes in the mceliece cryptosystem. Em *IEEE International Symposium on Information Theory (ISIT 2000)*, página 215, 2000. 2, 15

[SS11] Damien Stehlé e Ron Steinfeld. Making ntru as secure as worst-case problems over ideal lattices. Em *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, páginas 27–47. Springer, 2011. 2